Improving Artificial-Immune-System-Based Computing by Exploiting Intrinsic Features of Computer Architectures

Yiqi Deng, Peter J. Bentley, Alvee Momshad
Dept. of Computer Science
University College London
London, United Kingdom
y.deng.11@ucl.ac.uk, p.bentley@ucl.ac.uk, momshad.alvee.13@ucl.ac.uk

***Abstract*—Biological systems have become highly significant for traditional computer architectures as examples of highly complex self-organizing systems that perform tasks in parallel with no centralized control. However, few researchers have compared the suitability of different computing approaches for the unique features of Artificial Immune Systems (AIS) when trying to introduce novel computing architectures, and few consider the practicality of their solutions for real world machine learning problems. We propose that the efficacy of AIS-based computing for tackling real world datasets can be improved by the exploitation of intrinsic features of computer architectures. This paper reviews and evaluates current existing implementation solutions for AIS on different computing paradigms and introduces the idea of "C Principles" and "A Principles". Three Artificial Immune Systems implemented on different architectures are compared using these principles to examine the possibility of improving AIS through taking advantage of intrinsic hardware features.**

Keywords—Artificial Immune Systems; Systemic Computing; Multi-threaded Computing

I. INTRODUCTION

Nature is full of examples of biological computation that adapt to their environments and show remarkable tolerance to damage or faults. The biological immune system, which protects the body from pathogens, has great pattern recognition capability that distinguishes between foreign cells entering the body (non-self) and self cells [1]. More and more immune-based models and techniques have emerged in order to solve complex computational or engineering problems during the last decade [2]. The attempts mainly focus on two parts: the algorithm itself and alternative hardware implementations [1]. In the aspect of hardware implementation, due to the mixed nature of AIS-based computing, traditional and unconventional computing paradigms have been put forward as AIS-based computing substrates [3-20]. Although increasing research has been conducted to investigate better versions of AIS and more suitable computing architectures to accommodate them [3-20], few have compared the suitability of different computing approaches for the unique features of AIS when trying to introduce novel computing architectures, and few consider the practicality of their solutions for solving real-world machine learning problems.

We propose that the efficacy of AIS based computing for tackling real world datasets can be improved by the exploitation of shared intrinsic features between computer architectures and algorithms. In this paper, we aim to identify the intrinsic features of computer architectures that might improve the efficacy of AIS based computing for tackling real world datasets.

The paper is organized in the following way: In the section of Background related literatures are reviewed and measuring principles for algorithm performance are put forward. Three architectures are selected as the substrate on which corresponding Artificial Immune Systems are developed in order to examine the possibility of improving AIS through taking advantage of intrinsic hardware features. Further experiments are then conducted on five classic benchmark datasets with analysis of results.

II. BACKGROUND

Inspired by the biological immune system, the Artificial Immune System (AIS) is an emerging computational intelligence paradigm, which shares features with evolutionary learning such as noise tolerance, adaptability and dynamics [1]. Currently, there is an increasing number of new hardware architectures specially designed in order to better cater to the intrinsic features of AIS. In this section we will focus on the different substrates used to accommodate AIS-based computing.

A. Field-Programmable-Gate-Array-Based (FPGA-Based) AIS

Among all the implementations of AIS, AIS implemented on FPGA is perhaps the most diversified one as the ways FPGAs are organized are largely different from each other. Despite that, there are generally two ways of implementing AIS on FPGA. One is using multiple FPGAs with each of them as a distributed computing unit. For example, the Bionode System developed by Greensted and Tyrrell contains thirty individual FPGAs, which can be configured to model the functionality of a cell, connected in a loosely coupled network [3]. Similar attempts have been seen on [4], [5] and [6]. The other solution is to use one single

FPGA as a central processing unit with multiple logic unit flows. For instance, Smith et al. proposed a hardware implementation of a novel evolutionary algorithm inspired by protein/substrate binding exploited in artificial immune networks to classify Parkinson's patient's response to a conventional figure copying task [7]. An intrinsic FPGA-based evolvable hardware platform which exhibits high tolerance to transient faults by making use of chemical signals was proposed by Liu et al. [8]. Bradley and Tyrrell put forward an immunotronic architecture which runs in real time hardware and continuously provide monitoring over a finite state machine architecture for errors is developed in [9]. Canham and Tyrrell developed a multi-layered hardware artificial immune system coupled with an embryonic array [10]. The system consists of an acquired layer of the immune system to monitor unusual system behavior, a non-learning innate layer to locate the fault and a homogeneous array of logic units – an embryonic array to provide fault avoidance.

B. Sensor Network based AIS

AIS is also implemented on sensor networks. The General Suppression Control Framework (GSCF) is a framework inspired by the suppression hypothesis of the immune discrimination theory[11]. The possibility to apply Dendritic Cell Algorithm (DCA) in the attack detection in sensor network is investigated in [12]. An Intrusion Detection System (IDS) framework to be applied in the wireless sensor network context is proposed in [13]. However, regardless of model differences, in the case of large scale networks, sufficient sensors are required, which has a negative impact on system scalability.

C. GPU-based AIS

GPUs are regarded as another potential substrate to implement AIS. Using GPUs, a novel parallel data clustering algorithm based on the artificial immune network aiNet is proposed to improve its efficiency by 10 times [14]. While enjoying good scalability, the flow of the algorithm is relatively complex. Similar work is described by [15] where a GPUAIS is put forward on NVIDIA GPU using a probabilistic elite antibody selection.

D. Cloud-based AIS

By analyzing the characteristics of current cloud computing, Yang et al. propose a comprehensive real-time network risk evaluation model for cloud computing based on the correspondence between the artificial immune system antibody and pathogen invasion [16]. The way the proposed model places multiple immune cells into the network to perceive contexts against the traditional network security approach, which relies on a single terminal to check the environment, indicates good concurrency of the model.

E. Swarm-robot-based AIS

Granuloma formation is applied to solve the 'anchoring' issue in swarm robotics as it acts as a healing mechanism, trying to prevent bacterial infections from infecting other cells and to contain the infection by attracting other cells such as macrophages and T-cells to move to the site of infection [17]. A model proposed by Ismail excels in the aspects of fault tolerance and self-repair. Nevertheless, compared to hardware which realizes concurrency via circuits such as GPU, the scalability of the proposed system is not very cost-efficient. A method of cooperative control (T-cell modelling) and selection of group behavior strategy (B-cell modelling) based on immune system in distributed autonomous robotic system is proposed where a robot is regarded as a B-cell, each environmental condition as an antigen, a behavior strategy as an antibody and control parameter as a T-cell [18]. Even though it benefits from good concurrency, the proposed system suffers from poor performance on self-repair.

F. von Neumann based AIS

Apart from the architectures reviewed above, the most prevailing hardware is traditional von Neumann architecture. An artificial immune network model named aiNet is presented in [19] and applied to reduce redundancy as well as spatial distribution. Campelo et al. put forward the real-coded clonal selection algorithm (RCSA) for use in electromagnetic design optimization. Some features of the algorithm, such as the number of clones, mutation range, and the fraction of the population selected each generation are discussed [20]. The role of negative selection in an artificial immune system (AIS) for network intrusion detection is investigated in [21]. Martelot introduced and developed the notion of artificial metabolism using a conventional computer to simulate systemic computation (a novel method of distributed natural computation) to create an organism which uses input data stream to grow [22]. While showing some unusually good abilities for fault tolerance, the von Neumann architecture based simulation suffered from slow speed and scalability problems.

G. Summary

This brief summary of some of the literature illustrates that each new AIS implementation induces new trade-offs in terms of speed, accuracy, scalability and more difficult-to-measure aspects such as ease of encoding the problem or programmability. It is clear that every new architecture may be better or worse suited to supporting different aspects of AIS; every AIS implemented on a different architecture requires subtle changes to its algorithm that may impact its efficacy in significant ways. The question is then, how to measure these differences?

III. Measuring AIS Architectures

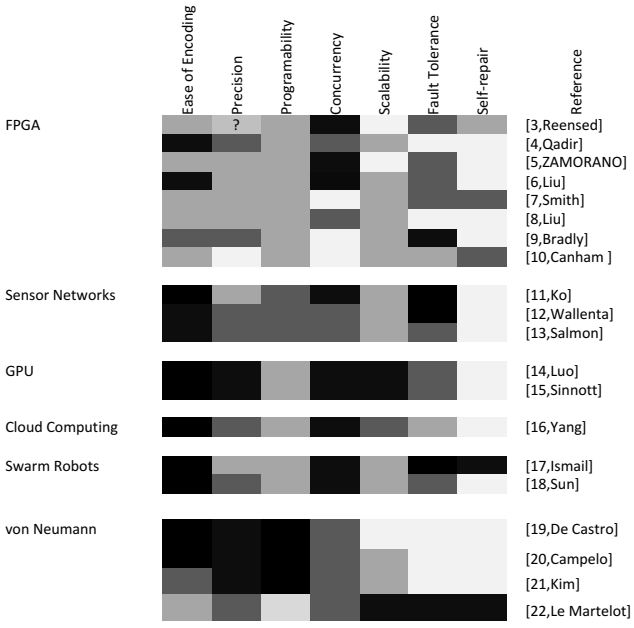
There are many indexes to evaluate the performance of architecture, just as there are many principles for evaluating performance of machine learning algorithms. However, few has performed direct comparisons between published AIS methods that use very different architectures with results for very different problems. In an attempt to enable some useful level of comparison, in this work, we introduce two types of evaluation principles for features specifically related to AIS: "capability principles" which are designed to be problem-independent and "accuracy principles" which require the algorithms to be compared on identical problems.

The seven capability principles (C Principles) used here are: 1. **Ease of encoding**: whether it is easy to for the user present the raw data to the algorithm. 2. **Precision**: the depth of knowledge encoded in data or the quality of presented data. In other words, the number of bits the algorithm could possibly process for each sample data. 3. **Programmability**: the difficulty of implementing the algorithm on the hardware

architecture. 4. **Concurrency**: to what extent are several computations executed simultaneously. 5. **Scalability**: the ease with which the platform hosting the algorithm can be modified, added or removed to accommodate the changing load and how effectively the algorithm can use increased computational resources. 6. **Fault-tolerance**: the capability for the algorithm to operate properly in the case of failure of one or more of the system components it based on. 7. **Self-repair**: the ability to repair/heal damage or corruption on the platform over time. As C Principles evaluate AIS methods on different architectures with results for various problems, the quantification relies on an analysis of the work and claims made by the authors of each paper. Each C Principle has four grades: Inadequate, Average, Above Average and Excellent. For each C Principle all the approaches reviewed are compared with each other. The grades are then determined by comparing to the one with the best grade and the one with the worst grade in this specific principle among all the reviewed approaches.

The five accuracy principles (A Principles) used in this work are: 1. **Accuracy**; 2. **Standard Deviation;** 3.**Processor Time;** 4. **Sensitivity;** 5. **Precision**.

Table 1 Summary of Evaluation using C Principles for reviewed architectures, where darker shades represent better, or more intrinsic, capabilities.



Using the capability principles, we can now assess the AIS systems published in the literature that were summarized in the previous section. Table 1 summarizes the reviewed architectures by evaluating based on the seven capability principles, with the black square denoting Excellent (more intrinsic to that architecture) and lightest grey as Inadequate (not intrinsic to that architecture). The grid square marked "?" indicates failure to evaluate due to absence of description by the authors of that work. Measures such as Fault tolerance and Self repair are included because this work anticipates the use of AIS for applications such as intrusion detection or robot control, where these additional capabilities of human immune systems are required in the AIS. For this work, a hypothetical "perfect AIS architecture" would have black squares denoting Excellent for all C principles.

It can be seen that no single hardware architecture shows complete superiority over the rest. GPU enjoys good performance in the aspect of concurrency and scalability at the cost of high demand on algorithm and code design as well as sacrifice on the feature of fault tolerance. Sensor networks and swarm robots show similar performance in the seven principles as both of them make use of distributed computing units interacting with each other with the only difference on the aspect of self-repair due to better mobility of swarm robot. For FPGA, the difference in the organization style is reflected on the disparate markings on the aspect of concurrency. The first four with higher marking all adopt the organizing style of multiple FPGAs, which as a result, show advantage in fault tolerance as well. Traditional von Neumann shows sharp imbalanced performance over the seven principles. It is worth mentioning that Le Martelot's unoptimized version of artificial immune system could serve as a good example of implementing right algorithms on the wrong hardware. On the other hand, different approaches of AIS also pose an impact on the performance of hardware.

Based on the analysis above, we believe that the efficacy of AIS based computing for tackling real world datasets can be improved by the exploitation of intrinsic features of computer architectures. In this rest of paper, we aim to understand to what extent the intrinsic features of computer architectures might improve the efficacy of AIS based computing.

IV. METHODOLOGY

The analysis of existing work in the previous section showed that each hardware platform shows advantages and disadvantages as measured by our capability principles. Here we focus on three architectures for AIS: traditional sequential von Neumann Architecture, Multi Threaded architecture, and Systemic Computation, that will enable us to explore how the capabilities of each may enhance the efficacy of a bio-inspired AIS algorithm. The von Neumann architecture is the standard computer architecture as used throughout the world; the multi-threaded architecture comprises parallel processes running on a modern multi-core processor.

Table 2 Comparison of VN, Multi-Threaded and SC

von Neumann Architecture	***Multi-Threaded***	***Systemic Computation***
Deterministic	Deterministic	Stochastic
Synchronous	Synchronous	Asynchronous
Centralised	Distributed	Distributed
Externally-organized	Externally-organized	Self-organized
Heterostatic	Heterostatic	Homoeostatic
Brittle	Robust	Robust
Limited	Limited	Autonomous
Fault intolerant	Fault-tolerant	Fault-tolerant

In contrast, Systemic Computation is a computer architecture designed to provide native support for common characteristics of biological processes yet still compatible with

current processors [23]. Instead of the traditional centralized view of computation, in SC all computation is distributed. There is no separation of data and code, or functionality into memory, ALU, and I/O. In SC, everything is regarded as a system. Systems could never be created, nor destroyed. Instead, discarded computation remnants are constantly transformed into new systems. Systems interact and differentiate with each other through a medium called "contextual" system which depicts what the two interacting systems transform into. Different contexts would result in various transformations accordingly. Another important concept in Systemic computation is scope. Systemic computation rules that interactions could only happen on the premise that both systems are in the same scope. Although it seems that natural computation such as SC shares more similarity in features with bio-inspired algorithms, for example AIS, it still remains a question whether Systemic Computation could be as fast as or even faster than traditional computing paradigm for such algorithms. To what extent, if any, could SC improve the performance on data? What is SC's scalability? These questions are made more pertinent because, like many radically different architectures, SC is most commonly used as a virtual machine running on a von Neumann architecture. Could there be any advantage at all in running an AIS on a virtual bio-inspired architecture, which itself runs on a traditional architecture? Would a multi-threaded architecture not provide a simpler and superior choice? To answer those questions, these three architectures will be directly compared using real world datasets.

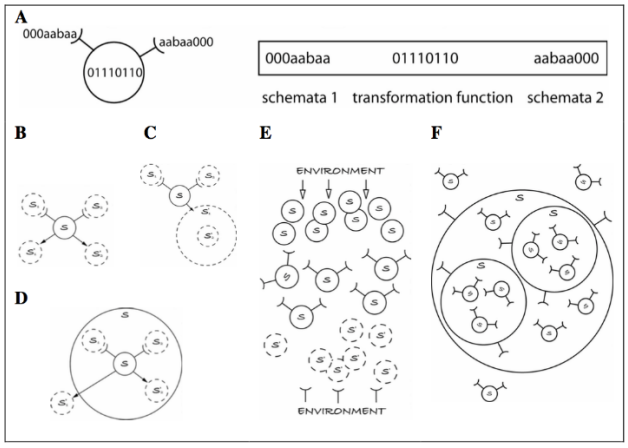


Figure 1 Systemic Computing relies on the concept of a system to perform all computation. A system comprises three elements: two schemata and a transformation function. Systems may be defined in memory as strings; they are graphically depicted as a circle surrounding the transformation function, with two "cups" or receptors representing the two schemata (A). The two schemata of a system define which other systems may match and hence be affected by this system. The transformation function of a system defines how two schemata-matching systems are changed when they interact with each other in the context of this system; arrows indicate transformed systems at time t+1 (B). A system may be pushed inside the scope of a second system through interaction with it (C). A system within the scope of a larger system may be pushed outside that scope through interaction with another system (D). Computation occurs by transforming input from the environment into systems, which interact with "organism" systems (equivalent to software or hardware) and potentially each other to produce new systems that provide output by stimulating or altering the environment (E). Most computation requires more structured systems enabling the equivalent of modules, subroutines, recursion and looping. Most or all systems may be active and capable of enabling interactions. Note the similarity between a typical systemic computation with the structure of biological systems such as the cell (F).[23]

A. AIS on von Neumann Architecture

For simplicity, the classical clonal selection algorithm CLOCLAS [24] is selected as below:

1. *Randomly generate an initial population of antibodies Ab. This is composed of two subsets Ab_m (memory population) and Ab_r (reservoir population).*
2. *Create a set of antigenic patterns Ag.*
3. *Select an antigen Ag_i from the population Ag.*
4. *For G generations*
   a) *For every member of the population Ab calculate its affinity to the antigen Ag_i.*
   b) *Select the n highest affinity antibodies and generate clones for each antibody in proportion to their affinity, placing the clones in a new population C^i.*
   c) *Mutate the clone population C^i to a degree inversely proportional to their affinity to produce a mature population C^{i*}.*
   d) *Re-calculate the affinity of each member in C^{i*} and select the highest score as candidate memory cell. If its affinity is greater than the current memory cell Ab_{m_i}, then the candidate becomes the new memory cell.*
   e) *Remove those antibodies with low affinity in the population Ab_r and replace them with new randomly generated members.*
5. *Select an antigen from the population to be classified* Ag^* *and Calculate the affinity of the antigen with each memory cell.*
6. *Set classification to the memory cell with highest affinity.*
7. *Loop until all antigens s have been presented.*

Here affinity function is the same as the one used in AIS-SC (see section 4.3).

B. AIS on Multi-threaded Architecture

The essence of multi-threaded architecture is to execute multiple processes or threads concurrently in a multi-core processor. As each antibody memory cell achieves learning independently from other memory cells [23], the AIS implemented on Multi-threaded Architecture executes M tasks in parallel where M is the number of memory cells (classes) in datasets. To assure this change, corresponding modification is needed for CLONCLAS [24] used on von Neumann Architecture

1.Each antibody memory only gets exposed to one antigen (training data) exclusively and each Antibody in the memory pool has its local copy of the reservoir pool.

2.For datasets with M classes, M antigens are selected each time with each from a different class and exposed simultaneously to corresponding antibody memories and reservoir pools.

3.The algorithms proceed such that if any particular class runs out of training data then from the next iteration only the remaining classes are picked up by the ExecutorService to be processed in parallel.

C. AIS on the Systemic Computation

Le Martelot first modelled an 'organism' using SC based on the notion of AIS [22]. To enable systemic computation model to be simulated using conventional computer processors, a new optimized systemic computation simulation has been created for this work, including the creation of a virtual architecture, instruction set, machine code and corresponding assembly language with compiler. Four implementation-specific features enable systemic computers to be tailored to a given application:

(i) the word-length / coding method,

(ii) the transformation function set / schemata matching method

(iii) the order of system interactions and

(iv) the scope definition method.

In the implementation described here, (i) in order to accommodate more information, schemata and transformation functions are defined by strings of 16 and 32 respectively, resulting in each system being 64 characters long; (ii) thirty-four transformation functions have been implemented; (iii) system interactions occur randomly except where a system is changed, in which case changed systems are chosen for subsequent interaction first; (iv) scopes are held globally in a system scope table. (Full details of SC can be found in [23].)

We also extend his early model and extend the capability of SC by introducing the idea of "virtual systems". Le Martelot's model uses one SC system to model each AIS 'cell'. However in our version of SC with virtual memory, each system points to a fixed and unique region of memory where a larger or more complex data structure can be held.

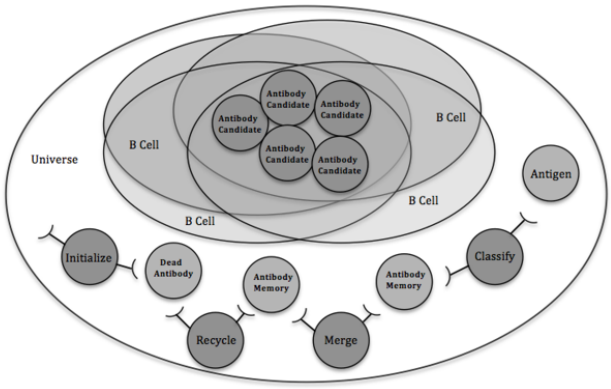


Figure 2 System organization of AIS-SC

The whole system is organized in the scope of a 'Universe system' which is an abstraction of the immune system. Inside it, B cells generate memory antibodies once attacked by training data (carried by antigens). The generated antibodies would then be secreted and used to recognize data with similar features (antigens).

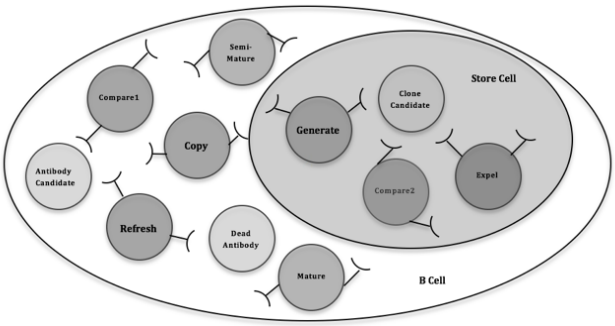


Figure 3 Zoom in Organization for B Cells

Bits 15 to bit 17 of transformation function in each SC system are used to represent types of data systems. The six types of data systems are: data (antigen), B cell, antibody candidate, clone antibody, antibody memory and winner cell. Bit 18 is used to identify the livelihood of systems: alive or dead /semi-finished or finished (see in Figure 3(a)).

```
#label data                 %b0000000000000101????????000000
#label semiwinnercell       %b0000000000000110????????000000
#label winnercell           %b0000000000000111????????000000

#label bcell                %b0000000000000001000???????000000
#label semifinibcell        %b0000000000000001010???????000000
#label finibcell            %b0000000000000001011???????000000

#label antibodycandidate    %b0000000000000001001???????000000
#label deadbodycandidate    %b0000000000000001000???????000000

#label clonecandidate       %b000000000000000011???????000000
#label bodymemory           %b0000000000000001111???????000000
```

```
#scope main
{
main
[0:107]bcell
[0:11]antigen
classifysys
suppresssys
}

#scope bcell0
{
bcell0
[0:647]antibodycandidate
storecell0
comparesys1
semimaturesys
maturesys
expelsys
refreshsys
}

#scope storecell0
{
storecell0
bcell0
comparesys2
}
```

Figure 4 SC System representation (a); Code for system organization (b)

After identifying the elementary systems, we move to the identification of contextual systems which are in charge of system interaction and transformation. An initialize context is implemented in order to vaccinate the B-cell with training data. In the first phase of maturation, the functionality of Compare1 system is streamlined as it only calculates the affinity between B-cell and antibody candidate and decides whether the affinity is better than the current winner whose corresponding index is stored in the right schemata of compare system.

$$Affinity(Ab_i, Ag_j) = \sqrt{\sum (Ab_i[k] - Ag_j[k])^2} \quad (1)$$

Should it succeed, the compare system hands the index of selected antibody candidate over to the chaining contextual system copy which stores the index to semi-winner cell. The

counter implemented in Compare1 system keeps track of continuous winning time for the current winner. When it reaches the counter threshold, the B-cell would be turn to semifinibcell. Correspondingly, semi-winner cell is transformed to winner cell by semi-mature context for next phase maturation. In the previous model [25], great efforts were wasted in going through each and every antibody candidate to find the final winner to proceed into next phase. By introducing the copy winner step to hold the selected winner, the successful interaction rate is improved more than 700 times.

Table 3 Summary of the Integrated AISO SC model functions

Function Name	Description
Initialize	Initializes a non-initialized data system with random values, transforms it to antibody candidate and inserts it to the first selection scope.
Compare 1 & 2	Both Compare 1 and 2 compare the distance between the antigen itself and the interacted antibody candidates and store better the index and distance in the right schema of the context. Both contexts also keep a counter of consecutive comparison time. In addition, Compare 2 expels the losing clone candidates to the general pool.
Mature	Transform the antigen to the state of second round of selection and mark the final selected antibody candidate as antibody memory.
Generate	Duplicates the winner antibody candidate with a mutation on each dimension proportional to its distance to the antigen it compares to.
Kill	Compares the distance between the two mutated duplicates antibody candidates and their corresponding antigens, mark the winner and record its winning time in the counter. If the counter exceeds counter threshold, transforms the winner to antibody memory.
Expel	Pulls the generated antibody memory out of the scope of the second selection to the root scope.
Merge	Calculates the distance between the two interacted antibody memories and if it is below the sigmoid threshold, pulls all the classified test data systems into one antibody memory and update the value of that memory with the average of the two memories. Meanwhile, transforms the other antibody memory into waste for recycle.
Classify	Decides whether the interacted test data systems belong to this category. If so, tags them.

Once semi-mature context finds matching system successfully, meaning the completion of first phase, the generate context chained after semi-mature context proliferate 5 mutated copies of the winner antibody candidates and absorbs each clone candidate it creates into the corresponding winner cell to produce antibody memories for the specific antigens later. Then Compare2 context follows similar rules of Compare1 with one more functionality to expel the losing clone candidates in the second comparison in to the general pool.

Moving into the next pair of interactions, Refresh system updates the loser antibody candidate with new randomized value to introduce new solutions to the system. Thus Refresh system acts as context, defining the refresh of failed data systems in the maturation. The mature context changes antibody winner in the second phase to antibody memory which is then expelled to the general pool by expel context. Finally, antigens in the environment are voted by the expelled antibody memories. More specifically, antigens will be classified in the class by which receives the most votes.

Moving to the scope: If we let each B cell hold a certain amount of antibody candidates exclusively as in reality, more than needed antibody candidates are set into each B cell considering the fact of system aging in order to cover the need for both comparison and clone generation, which is a waste both in storage resources and computing efficiency. In current model, all the B cells share a large antibody candidates pool. In other words, each B cell 'sees' all the antibody candidates at the very first beginning.

However, each B cell still holds private winner cell. By doing so, less antibody candidates are required and high rate of successful memory production is achieved. Another change is the scope relation settings between B cells and their corresponding winner cells. B cells and their corresponding winner cells are set to stay within each other's scope, namely, B cells could see their winner cells and so do winner cells. Consequently, the first comparisons happen in the scope of B cells with winner cells and general candidate pool while the second comparisons happen within the scope of winner cells with B cells and clone candidates.

V. Experiments

In order to understand how the performance of the models are affected by the substrate they are based on, the three models are assessed in terms of the C principles and A principles. To test the latter, all three models are compared on five benchmark datasets: Iris, Wine, Ecoli, Liver Disorder and Breast Cancer [26].

Two classes (imL and imS) from the Ecoli Dataset are not included in the experiments due to insufficient data. 20% of data in each dataset is reserved as test dataset while 10 fold cross validation is performed on the other 80% data to select the best parameter settings. For simplicity, the performance is evaluated by average accuracy, standard deviation of 10 runs, processor time, sensitivity and prevision for each class in all five datasets. The results are shown in Table 4.

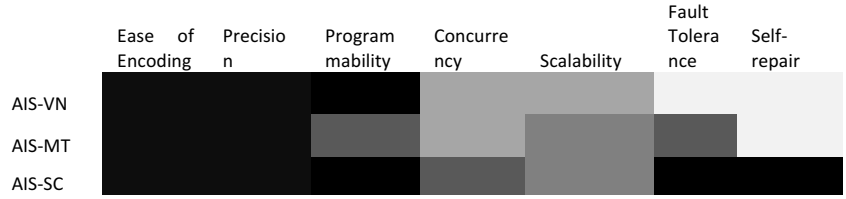
As shown in Table 4, in terms of average accuracy, the three models perform roughly the same. Although AIS-SC scores a little lower in Wine, Ecoli and Breast Cancer dataset. Its average

Table 4 Results on five benchmark datasets, where S= Sensitivity and P = Precision.

	Average Accuracy			Std			Processor time		
	AIS-VN	AIS-MT	AIS-SC	AIS-VN	AIS-MT	AIS-SC	AIS-VN	AIS-MT	AIS-SC
Iris	91.67%	88.07%	93.12%	4.77%	8.59%	4.91%	2.98s	2.32s	2.47s
Wine	79.91%	85.02%	77.72%	12.16%	7.37%	10.37%	46.18s	37.57s	32.31s
Ecoli	76.82%	84.60%	71.88%	7.71%	4.67%	8.77%	2.27s	2.07s	1.91s
Liver Disorder	52.52%	52.80%	63.71%	6.24%	5.23%	4.21%	23.67s	22.36s	22.59s
Breast Cancer	95.37%	91.30%	91.59%	2.75%	2.46%	3.24%	29.52s	26.19	24.97s

		Iris			Wine			Ecoli					Liver Disorder		Breast Cancer	
		Setosa	Versicolour	Virginia	Wine 1	Wine 2	Wine 3	CP	IM	PP	IMU	OM	Liver1	Liver2	Benign	Malignant
AIS-VN	S	100.0%	80.00%	90.00%	91.67%	57.14%	100.0%	82.76%	73.33%	72.51%	80.27%	66.26%	44.83%	60.00%	95.51%	80.85%
	P	100.0%	88.89%	81.82%	91.67%	88.89%	90.91%	96.00%	73.33%	78.69%	70.00%	90.51%	44.83%	60.00%	90.43%	90.48%
AIS-MT	S	100.0%	78.77%	85.27%	93.22%	65.27%	99.45%	87.26%	76.48%	73.52%	90.77%	70.21%	45.13%	59.87%	93.45%	80.88%
	P	100.0%	86.88%	82.34%	94.44%	60.51%	95.77%	94.01%	72.88%	74.49%	80.11%	83.33%	47.13%	60.00%	89.16%	85.23%
AIS-SC	S	100.0%	87.29%	88.53%	89.67%	60.11%	95.17%	81.21%	71.54%	71.15%	82.27%	64.96%	65.66%	62.11%	93.48%	82.74%
	P	100.0%	89.77%	88.99%	91.67%	85.33%	92.17%	94.00%	73.93%	75.55%	77.97%	85.29%	69.37%	70.23%	89.55%	89.19%

Table 5 Evaluation of three methods using the C principles



accuracy in Liver Disorder dataset is as much as 11.2% higher than that of AIS-VN, 10.9% higher than AIS-MT and 1.5% higher in Iris Dataset, 5.1% higher than AIS-MT. For all five datasets, AIS-SC has a more balanced performance in not only general as a whole, but sensitivity and precision across classes in all five datasets as well, compared to AIS implemented on other two architectures. In the aspect of calculating speed, there is a clear benefit from the parallel nature of Systemic Computing, with AIS-SC performing quicker in all the five datasets. The difference is particularly obvious for Wine and Breast Cancer datasets (nearly 1.7 times quicker than AIS-VN) which shows AIS-SC is much more competitive in dealing with high dimension data. (This speedup occurs despite the fact that SC is running as an optimized virtual machine on a von Neumann computer, and is not truly executing in parallel.) As for standard deviation, there is no clear winner across the three implementations. AIS-SC shows a slightly higher standard deviation, which is largely due to the fact that Systemic Computing is highly stochastic.

In summary, the results using the A principles show that the same AIS algorithm implemented on different computing architectures can produce subtly different results, because of the changes necessary to implement that algorithm on differing substrates. Considering the C principles, in terms of data presentation which is mainly illustrated by ease of encoding and precision, AIS implemented on all three selected architectures show equivalent performance. As for programmability, as shown above, AIS-SC is also quite intuitive while programming on VN is commonly accepted by researchers. On the other hand, AIS-MT is more demanding in programming skills. Furthermore, the innate mechanism endows AIS-SC with higher capabilities in Concurrency, Fault Tolerance and Self-repair especially while the other two versions fails to demonstrate advantage in the above three aspects. Table 5 shows the three versions of the AIS method evaluated using the C principles.

From the table above it is clear to see that AIS-MT improves C Principles in the aspect of Scalability and Fault-Tolerance and AIS-SC shows further improvement in the four aspects which AIS-VN is not good at by exploiting the common innate features between Multi-Threaded Architecture, Systemic Computer and AIS. As AIS-SC is running on a virtual machine on VN architecture, further improvement is expected on a true SC computer [23], which will be examined in future experiments.

VI. Conclusion

In this paper we reviewed the development on traditional and bio-inspired algorithms and evaluated current existing implementation solutions for AIS both on conventional computing architecture and unconventional computing paradigms. The work introduced the concept of C Principles and A Principles in order to compare the efficacy of AIS based computing in terms of the exploitation of shared intrinsic features between computer architectures and algorithms. The work then focused on three computing architectures: von Neumann Architecture, Multi-threaded Architecture and Systemic Computing. We examined how to best realize AIS on

a non von Neumann computing architecture by improving the classification system inspired from AIS on the platform of systemic computation. AIS-VN, AIS-MT and AIS-SC were compared in terms of C Principles and A Principles, the latter using five benchmark datasets. The results showed that the use of the bio-inspired SC architecture, even when running as a virtual machine, improved the capability of the algorithm in terms of scalability and fault tolerance while also improving processing time, and achieving a competitive accuracy rate. This provides evidence that measurable performance gains can be achieved if hardware architectures are designed to support bio-inspired algorithms such as AIS by ensuring that both share intrinsic features.

REFERENCES

[1] DE CASTRO, L. N. 2007. Fundamentals of natural computing: an overview. *Physics of Life Reviews,* 4**,** 1-36.

[2] DE CASTRO, L. N., VON ZUBEN, F. J. & KNIDEL, H. 2007. *Artificial Immune Systems*, Springer.

[3] REENSTED, A. J. & TYRRELL, A. M. An endocrinologic-inspired hardware implementation of a multicellular system. Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on, 2004. IEEE, 245-252.

[4] QADIR, O., LIU, J., TIMMIS, J., TEMPESTI, G. & TYRRELL, A. Hardware architecture for a bidirectional hetero-associative protein processing associative memory. Evolutionary Computation (CEC), 2011 IEEE Congress on, 2011. IEEE, 208-215.

[5] ZAMORANO, A. G., TIMMIS, J. & TYRRELL, A. A flexible decentralised communication architecture on a field programmable gate array for swarm system simulations. Evolutionary Computation (CEC), 2011 IEEE Congress on, 2011. IEEE, 230-237.

[6] LIU, Y. & LI, W. Study on Hardware Implementation of Artificial Immune System. Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference on, 2010. IEEE, 1-4.

[7] SMITH, S. L., GREENSTED, A. & TIMMIS, J. 2008. Hardware acceleration of an immune network inspired evolutionary algorithm for medical diagnosis. *Evolvable Systems: From Biology to Hardware.* Springer.

[8] LIU, H., MILLER, J. F. & TYRRELL, A. M. Intrinsic evolvable hardware implementation of a robust biological development model for digital systems. Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on, 2005. IEEE, 87-92.

[9] BRADLEY, D. & TYRRELL, A. A hardware immune system for benchmark state machine error detection. wcci, 2002. IEEE, 813-818.

[10] CANHAM, R. & TYRRELL, A. M. 2003. A learning, multi-layered, hardware artificial immune system implemented upon an embryonic array. *Evolvable Systems: From Biology to Hardware.* Springer.

[11] KO, A., LAU, H. Y. K. & LEE, N. M. Y. 2008. AIS based distributed wireless sensor network for mobile search and rescue robot tracking. *Artificial Immune Systems.* Springer.

[12] WALLENTA, C., KIM, J., BENTLEY, P. J. & HAILES, S. 2010. Detecting interest cache poisoning in sensor networks using an artificial immune algorithm. Applied Intelligence, 32, 1-26.

[13] SALMON, H. M., DE FARIAS, C. M., LOUREIRO, P., PIRMEZ, L., ROSSETTO, S., RODRIGUES, P. H. D. A., PIRMEZ, R., DELICATO, F. C. & DA COSTA CARMO, L. F. R. 2013. Intrusion detection system for wireless sensor networks using danger theory immune-inspired techniques. *International journal of wireless information networks,* 20**,** 39-66.

[14] LUO, R. & YIN, Q. 2011. A novel parallel clustering algorithm based on artificial immune network using nVidia CUDA framework. *Human-Computer Interaction. Design and Development Approaches.* Springer.

[15] SINNOTT-ARMSTRONG, N. A., GRANIZO-MACKENZIE, D. & MOORE, J. H. 2010. High performance parallel disease detection: an artificial immune system for graphics processing units. *Computational Genetics Laboratory Dartmouth Medical SchoolLebanon, NH,* 3756.

[16] YANG, J., WANG, C., LIU, C. & YU, L. 2013. Cloud computing for network security intrusion detection system. *Journal of Networks,* 8**,** 140-147.

[17] ISMAIL, A. R. 2011. Immune-inspired self-healing swarm robotic systems.

[18] SUN, S.-J., LEE, D.-W. & SIM, K.-B. Artificial immune-based swarm behaviors of distributed autonomous robotic systems. Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, 2001. IEEE, 3993-3998.

[19] DE CASTRO, L. N. & VON ZUBEN, F. J. 2001. aiNet: an artificial immune network for data analysis. Data mining: a heuristic approach, 1, 231-259.

[20] CAMPELO, F., GUIMARÃES, F. G., IGARASHI, H. & RAMÍREZ, J. 2005. A clonal selection algorithm for optimization in electromagnetics. *Magnetics, IEEE Transactions on,* 41**,** 1736-1739.

[21] KIM, J. & BENTLEY, P. J. An evaluation of negative selection in an artificial immune system for network intrusion detection. Proceedings of GECCO, 2001. 1330-1337.

[22] LE MARTELOT, E. 2010. Investigating and Analysing Natural Properties Enabled by Systemic Computation within Nature-inspired Computer Models. *EngD Thesis, Department of Electrical & Electronic Engineering, UCL, London***,** 363.

[23] Bentley, P. J. (2009) Methods for Improving Simulations of Biological Systems: Systemic Computation and Fractal Proteins. In Special Issue on Synthetic Biology, J R Soc Interface 2009 6:S451-S466; doi:10.1098/rsif.2008.0505.focus.

[24] White, J.A. and Garrett, S.M., 2003, September. Improved pattern recognition with artificial clonal selection?. In International Conference on Artificial Immune Systems (pp. 181-193). Springer Berlin Heidelberg.

[25] DENG, Y. & BENTLEY, P. J. Dynamic learning of heart sounds with changing noise: an AIS-based multi-agent model using systemic computation. Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion, 2014. ACM, 985-992.

[26] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.