Entropy **2015**, *17*, 1-x; doi:10.3390/——

OPEN ACCESS
entropy
ISSN 1099-4300
www.mdpi.com/journal/entropy

Article

Topological Characterization of Complex Systems: Using Persistent Entropy

Emanuela Merelli [1,*], Matteo Rucco [1], Peter Sloot [2,3,4] and Luca Tesei [1]

[1] School of Science and Technology, University of Camerino, Camerino 62032, Italy;
E-Mails: ruccomatteo@gmail.com (M.R.); luca.tesei@unicam.it (L.T.)
[2] University of Amsterdam, Amsterdam 1098 XH, The Netherlands; E-Mail: p.m.a.sloot@uva.nl
[3] Complexity Institute, NTU, Singapore 637723, Singapore
[4] ITMO University, St. Petersburg 199034, Russian

* Author to whom correspondence should be addressed; E-Mail: emanuela.merelli@unicam.it;
Tel.: +39 0737 402567

Academic Editor: Raúl Alcaraz Martínez

Received: 28 July 2015 / Accepted: 29 September 2015 / Published: xx

---

Abstract: In this paper, we propose a methodology for deriving a model of a complex system by exploiting the information extracted from topological data analysis. Central to our approach is the $S[B]$ paradigm in which a complex system is represented by a two-level model. One level, the structural S one, is derived using the newly-introduced quantitative concept of persistent entropy, and it is described by a persistent entropy automaton. The other level, the behavioral B one, is characterized by a network of interacting computational agents. The presented methodology is applied to a real case study, the idiotypic network of the mammalian immune system.

Keywords: topological data analysis; persistent entropy automaton; higher dimensional automata; immune system; idiotypic network; computational agents

---

1. Introduction

Complex systems are typically characterized by a finite, usually large, number of non-identical interacting entities that often are complex systems themselves, with strategies and autonomous behaviors. Complex system science is a relatively young research area, and it is raising interest among

many researchers, mainly thanks to the emerging of new techniques in several fields, such as physics, mathematics, computer science and data analysis [1].

Complex systems are analyzed using mainly two different approaches: one that provides a global and abstract description by systems of differential equations [2] and the other that focuses on the interacting components, which are generally simulated by an agent-based model and simulation framework [3].

In terms of data volume, a complex system can be associated with a big dataset. To extract useful information from these data, new techniques have been recently introduced, most of them in the area of topological data analysis (TDA) [4]. Topology is the branch of mathematics that studies the connectivity of spaces or, in general, the features of shapes [5]. TDA is largely used for the analysis of complex systems; for instance, Ibekwe *et al.* [6] used TDA for reconstructing the relationship structure of *Escherichia coli* O157, and the authors proved that the non-O157 is in 32 soils (16 organic and 16 conventionally-managed soils). TDA was also used by De Silva [7] for the analysis of sensor networks, and it was successfully applied to the study of viral evolution in biological complex systems [8]. Petri *et al.* [9] used a homological approach for studying the characteristics of functional brain networks at the mesoscopic level.

TDA is a new discipline inspired by homology theory [5]. Informally, homology is a machinery for counting the number of n-dimensional holes in a topological space. In our approach, we consider only combinatorial objects: simplices (*i.e.*, vertices, line segments, triangles, tetrahedra, and so on) and simplicial complexes, *i.e.*, collections of simplices [5] (see Figure 1). Simplicial complexes are analyzed via persistent homology, which allows one to identify the generators of persistent n-dimensional holes [4,10].

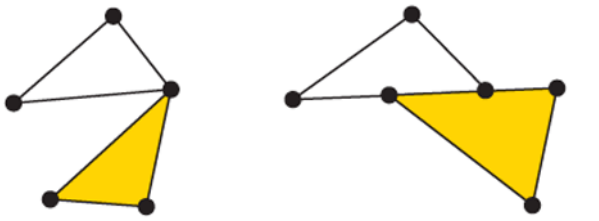


Figure 1. On the left, a simplicial complex formed by one two-simplex (yellow filled triangle) and three one-simplices (segments forming the non-filled triangle) and three zero-simplices (the vertices). On the right, an invalid simplicial complex; the intersection between the yellow filled triangle and the one-simplex belonging to the upper triangle is not empty, and they do not share a common face.

In this work, we propose a methodology for modeling complex systems that is based on TDA. Our contribution has been inspired by $S[B]$ [11], a paradigm for modeling complex systems. It was used as a general framework for modeling adaptivity in software systems [12], but it was conceived for all kinds of complex systems (biological, physical, chemical, and so on).

Figure 2 graphically shows our methodology. It is an iterative process where each iteration corresponds to the analysis of a data sample (observation) of the complex system under study. In each

iteration, data are first converted into a weighted graph in order to apply the subsequent steps. The graph is taken as the scaffold of the topological space from which the representation with a filtered simplicial complex is derived. Then, persistent homology is computed in order to calculate the topological invariants of the underlying topological space and to identify their generators. At this point, two kinds of information are derived: the computational agents (from the generators) that give rise to the behavioral level B of $S[B]$ and the persistent entropy (from the invariants) that is a newly-introduced entropic measure describing the global dynamics of the complex system.

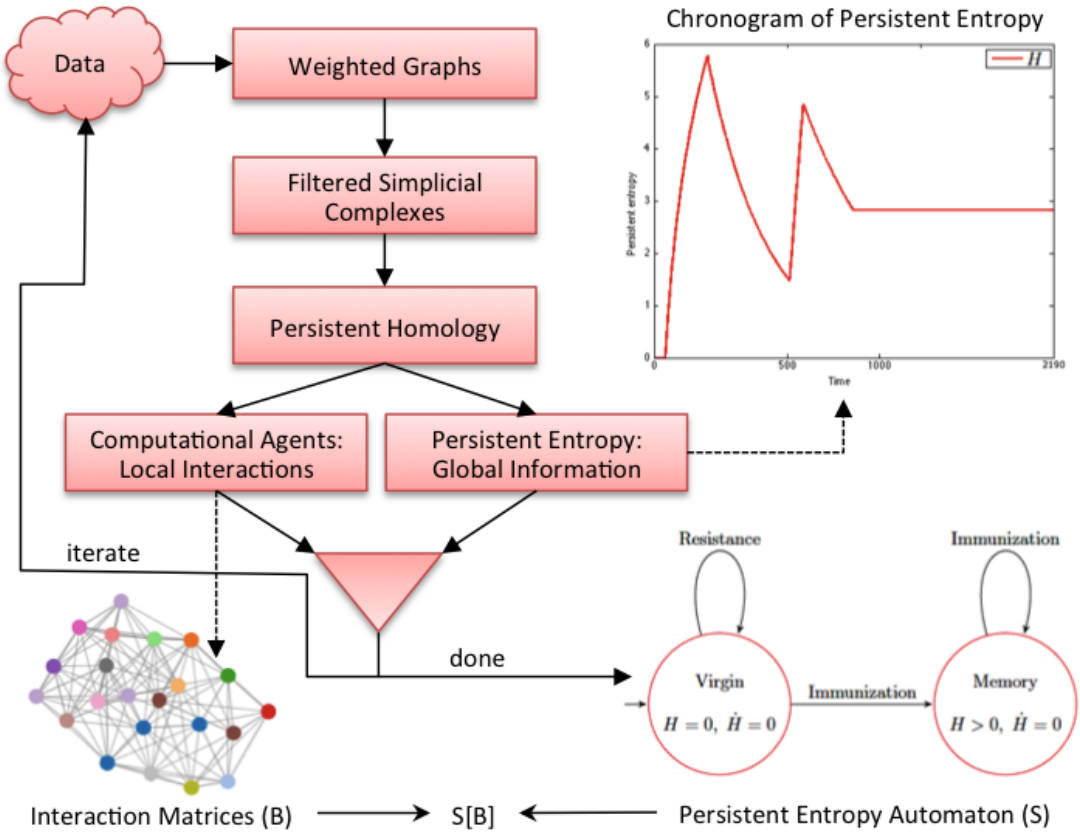


Figure 2. Graphical representation of our methodology. All of the details are explained in the text.

At the end of the iteration, we construct an $S[B]$ model in which the structural level S is a persistent entropy automaton, derived from all of the values of persistent entropy calculated during the iterations. The computational agents identified at each iteration step, formally represented by interaction matrices, are the sequence of B levels associated with the observed evolution of $S[B]$.

To validate our methodology, we consider, as a case study, the idiotypic network (IN) model [13] of the mammalian immune system. The latter has been largely studied as a complex adaptive system [14], and several other models are available. Concerning the IN model, we consider the simplified description given by Parisi [15], and we use the agent-based simulator *C-ImmSim* [16] for generating the data to validate our approach. In this work, we analyzed simulated data instead of real data in order to validate the correctness of the methodology in a more controlled way.

The paper is organized as follows. Section 2 introduces the case study of the IN. Section 3 describes our methodology step by step in order to extract from that data a two-level model according to the

$S[B]$ paradigm. In Section 4, the methodology is applied to the case study of IN, and finally, Section 5 provides concluding remarks.

2. Case Study

In this paper, we illustrate the steps of the proposed methodology applied to a case study in a biological immune system, specifically to the immune network theory by Nielse Jerne [13]. Jerne theorized that the immune system can be thought of as a regulated network of antibodies and anti-antibodies, called the idiotypic network. Let us briefly recall the main mechanisms described by the model. When an antigen is presented to the organism, the immune system reacts in the following two ways: innate immunity and adaptive (or acquired) immunity. Innate immune defenses are non-specific, meaning that these systems respond to pathogens in a generic way. The innate immune system is the dominant system of the host defense in most organisms; it involves the epithelial barriers, the phagocytes, the dendritic cells, the plasma proteins and the NK cells. The adaptive immune system, on the other hand, is called into action against pathogens that are able to evade or overcome innate immune defenses. The first reaction of the immune system is governed by the mechanism of the innate immunity, and its lifespan is up to 12 h; after that period, in the case of the presence of antigens, the adaptive response is triggered. The adaptive immune system is mainly composed of B cells, antibodies, naive T cells and effector T cells. When activated, these components adapt to the presence of infectious agents by activating, proliferating and creating potent mechanisms for neutralizing or eliminating the antigens. The lifespan of the adaptive response depends on the type of infection. In this work, we are interested in studying the behavior of the antibodies involved during the adaptive response, and we present a general example of this scenario. Suppose an antigen is recognized by B cells, which secrete antibodies, say Ab_1. Ab_1 themselves are then recognized as anti-antibodies by anti-idiotypic B cells, which secrete other antibodies, say Ab_2. Thus, further interactions lead to Ab_3 antibodies that recognize the Ab_2 ones, and so on.

In an IN, there is no intrinsic difference between an antigen and an antibody. Moreover, any node of the network can bind to and be bound to any other antigen or antibody. This phenomenon is known as the idiotypic cascade. During the ontogenesis phase, the immune system learns which antibodies should not be produced, and the system remembers these decisions for its entire life. This phenomenon is called immunological memory. The important properties of this system, including memory, are then the properties of the network of cells as a whole, rather than of the individual cells [17]. The IN model has been used by Parisi [15] to derive a simplified model for describing the dynamics of a functional network of antibodies in the absence of antigens. The dynamics h_i of each antibody i ($i = 1, 2, \ldots n$) at equilibrium is simply described by Equation (1):

$$h_i = \mathcal{S} + \sum_{k=1}^{n} J_{i,k} c_k \tag{1}$$

where $J_{i,k}$ ($J_{i,i} = 0; J_{i,k} = J_{k,i}$) represents the influence of antibody k on antibody i. If $J_{i,k}$ is positive, antibody k triggers the production of antibody i, whereas if $J_{i,k}$ is negative, antibody k suppresses the production of antibody i. $|J_{i,k}|$ is a measure of the efficiency of the control of antibody k on antibody i. The values of $J_{i,k}$ are distributed in the interval $[-1; +1]$. $\mathcal{S}$ is a threshold parameter that regulates the dynamics when the couplings $J_{i,k}$ are all very small; otherwise, one can assume $\mathcal{S}$ equal to zero.

The concentration c_i of antibody i is assumed to have, in absence of external antigens, only two values, conventionally zero or one (in the presence of antigen concentrations, c_i might become ≥ 1). The immune system state is determined by the values of all c_i's for all possible antibodies $(i = 1, 2, \ldots, n)$. h_i represents the total stimulatory/inhibitory effect (depending on its sign) of the whole network on the i-th antibody. h_i is positive when the excitatory effect of the other antibodies is greater than the suppressive effect; otherwise, h_i is negative.

3. Methodology

The aim of our methodology is to extract, applying TDA, local and global information from a dataset. In particular, the output of the methodology is a set of models specified within the $S[B]$ paradigm [11]. In the $S[B]$ paradigm, a model has two levels of description, namely the S global or structural level and the B local or behavioral level, which are entangled in order to express the behavior of the system as a whole. The S level describes how the system evolves following global information coming from the environment in which it is operating and from the interactions and the evolutions of the entities of the B level. Note that this approach has some similarities to hierarchical models like the hierarchical automata proposed by Mikk *et al.* [18]. However, in our case, the levels cannot be seen as just different levels of abstraction, but they can exist only in the entangled $S[B]$ model.

3.1. From Data to Weighted Graphs

We start from data that comes from observations, over time, of the complex system under study. The first step to be accomplished is to represent such data as weighted graphs where:

- nodes represent the interacting components;
- an edge, equipped with a weight, expresses an interaction (or a distance) between two components.

Note that this step is always possible because the weight can be defined using statistical descriptors (correlation coefficients, scoring systems, and so on), metric functions (normalized Euclidean distance, Hamming distance, and so on) or domain-dependent features.

3.2. From Weighted Graphs to Filtered Simplicial Complexes

In this work, we are interested in filtering simplicial complexes for studying the connectivity among the entities belonging to a complex system. The data space can be represented by simplicial complexes, which are combinatorial objects. From a geometrical point of view, a simplicial complex is obtained by properly gluing together smaller components, the so-called simplices. Low dimensional simplices are:

- 0-simplex, geometrically represented by a vertex;
- 1-simplex, geometrically represented by an edge;
- 2-simplex, geometrically represented by a filled triangle;
- 3-simplex, geometrically represented by a filled tetrahedron formed by filled triangles;
- ...

In order to build a simplicial complex, the intersection between two simplices must be empty or they must have in common at least one face (see Figure 1). Graphs are strictly related to the representation of topological spaces; in particular, a graph can be viewed as a simplicial complex obtained by gluing together 0-simplices through 1-simplices.

Given a directed or undirected graph, it is possible to construct from it a simplicial complex following several approaches [19]. In this work, we use the clique complexes; given a graph $G = (V, E)$, a clique is a complete subgraph, and a maximal clique is a clique that is not contained in a larger clique. We use the Bron–Kerbosch algorithm [20] for listing all of the maximal cliques of a graph. As shown in Figure 3, a simplicial clique complex of G is the simplicial complex whose simplices are all of the maximal cliques of G. Since a k-clique is equivalent to a $(k-1)$-simplicial complex, then a 3-clique is equivalent to a 2-simplicial complex, and so on [21].

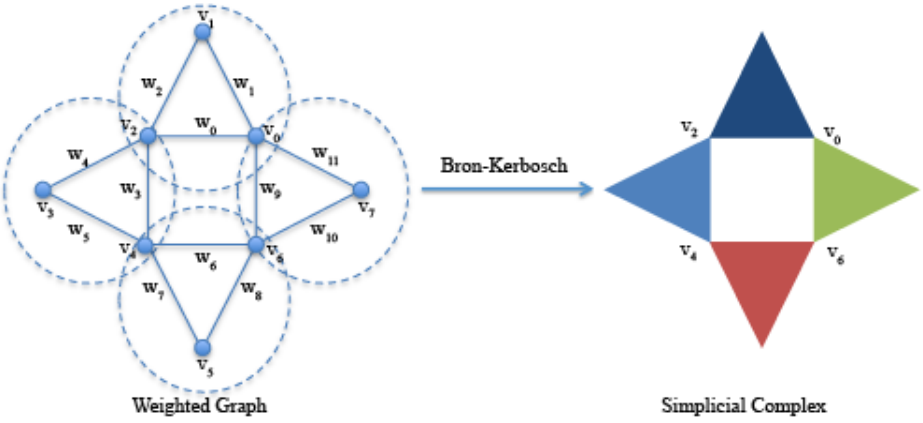


Figure 3. On the left, an undirected graph formed by eight vertices $V = \{v_0, v_1, \ldots, v_7\}$ and twelve edges with weights $W = \{w_0, w_1, \ldots, w_{11}\}$ with $w_0 \leq w_1 \cdots \leq w_{11}$. Highlighted by dashed circles, the four 3-maximal cliques $\{v_0, v_1, v_2\}$, $\{v_2, v_3, v_4\}$, $\{v_4, v_5, v_6\}$ and $\{v_6, v_7, v_0\}$ identified by the Bron–Kerbosch algorithm. On the right, the simplicial complex formed by the four 2-simplices (filled triangles) corresponding to the four 3-maximal cliques.

The computational algorithms for the characterization of simplicial complexes require as input a filtered simplicial complex, *i.e.*, a simplicial complex equipped with a filter value. A filter value [22] can be considered as a time-appearing value along the process described below. The set of obtained filter values is called the filter set and must be a strictly ordered set. The general algorithm for building a filtered simplicial clique complex from a weighted graph is described as follows:

1. list all maximal cliques of G with the Bron–Kerbosch algorithm;
2. for each maximal clique, select the minimum (or the maximum) value of the weights of its edges;
3. for each maximal clique, assign the value selected in Equation (2) to the corresponding simplicial complex as the filter value.

In Figure 4, the algorithm is applied to the graph in Figure 3.

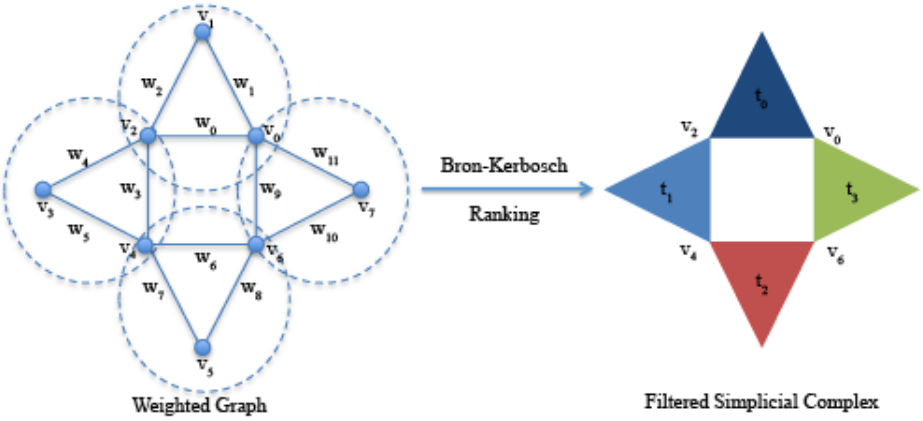


Figure 4. On the right, the filtered simplicial complex derived from the algorithm where the filter values are selected as the minimum of the weights. As a result, we set $t_0 = w_0$, $t_1 = w_3, t_2 = w_6, t_3 = w_9$, because we want a filter set $F = \{t_0, t_1, t_2, t_3\}$, such that $t_0 < t_1 < t_2 < t_3$.

3.3. Computation of Persistent Homology on Filtered Simplicial Complexes

Topological spaces can be characterized by using topological invariants. In this work, we use simplicial complexes as topological spaces and persistent homology to determine their invariants [5]. Persistent homology is the computational implementation of homology, which allows one to describe a simplicial complex in terms of n-dimensional holes. These quantities are expressed by the Betti numbers β_i, where $i \geq 0, i \in \mathbb{N}$, described as follows:

- β_0 corresponds to the number of connected components;
- β_1 corresponds to the number of planar holes;
- β_2 corresponds to the number of voids in solid objects (2-dimensional holes);
- ...

Moreover, for Betti numbers $\beta_i > 0$, a set $\mathcal{V}_i$ of generators of the corresponding topological features can be identified. Such generators are the simplices involved in the formation of the i-dimensional holes. Elements of $\mathcal{V}_0$ will be denoted by $[v_0], [v_1], \ldots$ to identify 0-simplices; elements of $\mathcal{V}_1$ will be denoted by edges $[v_0, v_1], [v_1, v_2], \ldots$; elements of $\mathcal{V}_2$ are (filled) triangles denoted by $[v_0, v_1, v_2], [v_0, v_1, v_3], \ldots$, and so on [5].

Basically, persistent homology is an incremental construction of the final filtered simplicial complex. The clique weight rank persistent homology algorithm (CWRPH) [23] is used, which has been efficiently implemented by the tool jHoles [24]. At each step of the algorithm, the family of simplices associated with the current filter value are introduced into the topological space, and the Betti numbers are computed. Figure 5 shows these steps applied to the filtered simplicial complex of Figure 4. The output of the algorithm is usually graphically represented by Betti barcodes. A Betti barcode, as shown in Figure 5, is a collection of line segments (or points in the case of persistent diagrams [4]), where each line (labeled `Dim 0` and `Dim 1`) is equipped with two pieces of information: the lifespan, in terms of appearance and disappearance, of a Betti number in the form of an interval $[a; b)$ and the corresponding set of generators.

If the line persists all over the procedure, *i.e.*, it survives also during the subsequent introduction of new simplices, it is called persistent, and $b = \infty$. Otherwise, it is classified as topological noise.

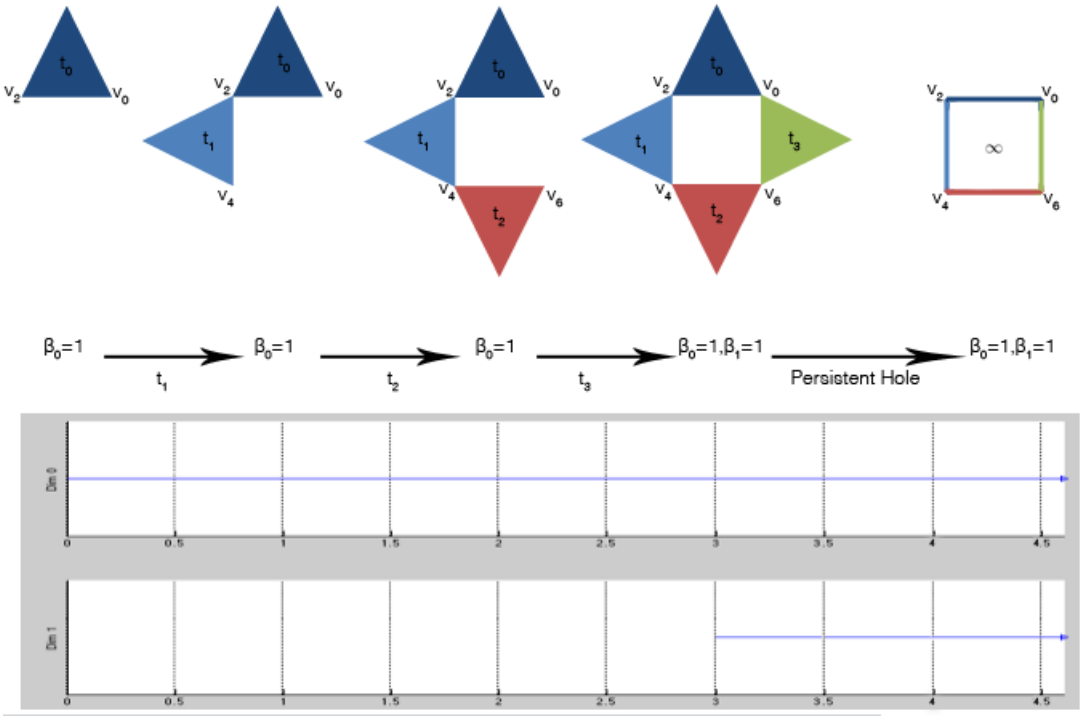


Figure 5. Computation of persistent homology over a filtered simplicial complex: at filter value t_0, the first 2-simplex (blue filled triangle) is introduced. The topological space is characterized by one connected component; the Betti numbers are $\beta_0 = 1$, $\beta_i = 0$ for all $i \geq 1$. These Betti numbers are the same until t_3 when the last 2-simplex (green filled triangle) is introduced, and it is connected to the others. For filter value t_3, the simplicial complex is still characterized by one connected component $\beta_0 = 1$, but also by one 1-dimensional hole, *i.e.*, $\beta_1 = 1$. The generator of the connected component is $\{[v_0]\}$, while the generators of the persistent hole are $\{[v_0, v_2], [v_2, v_4], [v_4, v_6], [v_6, v_0]\}$.

In Figure 5, a connected component appeared at filter value t_0, and its set of generators is $\{[v_0]\}$; it is persistent, as indicated in the `Dim 0` line of the Betti barcode. Similarly, a 1-dimensional hole appeared at filter value t_3, and its set of generators is $\{[v_0, v_2], [v_2, v_4], [v_4, v_6], [v_6, v_0]\}$; it is persistent, as indicated in the `Dim 1` line of the Betti barcode. Note that in this example, all of the lines in the barcode are persistent. However, this is not the case in general, because topological noise can be present.

At this point, we need to identify the components of the complex system interacting with each other and with the environment in which the system is immersed. Moreover, we need to characterize the latter one. Thus, from persistent homology, the methodology independently derives: (1) a set of computational agents together with their interaction matrices; and (2) the persistent entropy as a global property of the environment.

3.4. From Persistent Homology to Computational Agents: Local Interactions

Each non-empty set of generators $\mathcal{V}_i$ represents $(i+1)$-body interactions among components, *i.e.*, the methodology is able to automatically extract some hidden relationships (2-, 3-, ..., n-relations) among system components. For instance, the persistent homology computed in Figure 5 has non-empty sets of generators $\mathcal{V}_0 = \{[v_0]\}$ and $\mathcal{V}_1 = \{[v_0, v_1], [v_2, v_4], [v_4, v_6], [v_6, v_0]\}$. $\mathcal{V}_0$, containing only one element, simply tells us that the data represent only one system. On the other hand, $\mathcal{V}_1$ identifies four possible 2-body interactions among four interacting components, for which the vertices v_0, v_2, v_4 and v_6 can be taken as representatives. In general, for each non-empty $\mathcal{V}_i$, an interaction matrix J_i is obtained, such that the components (represented by vertices) can realize a multiple interaction of degree $i+1$. Note that this is a simple case of a more general model introduced by the authors in [25], which can be associated with the data space.

Since we want to derive a computational model for the complex system under study, it is very natural to use the concept of computational agent [2,3] to represent the identified interacting components. An agent can execute a generic number of actions, possibly concurrently. However, TDA extracts information about connectivity among agents, but it does not provide any information regarding the agent behavior. Thus, for obtaining a complete operational model of the interactions, there is still the need to use domain-based knowledge. For instance, in Section 4, we will use the Jerne model described in Section 2 with higher dimensional automata [26] as the computational model.

3.5. From Persistent Homology to Persistent Entropy: Global Information

In order to exploit global information from TDA, we use the notion of persistent entropy, introduced in [27], to characterize the environment. This entropy measure is basically calculated using the persistent Betti barcodes. By definition, the value of the persistent entropy is strongly related to the topological structures derived from the data. Given the maximum filtration value m of the set F and the set I of lines in a Betti barcode, the persistent entropy H of the topological space is calculated as follows:

$$H = -\sum_{i \in I} p_i log(p_i)$$

where $p_i = \frac{l_i}{L}$, $l_i = b_i - a_i$ and $L = \sum_{i \in I} l_i$. Note that, when topological noise is present, for each dimension of the Betti barcode, there can be more than one line, denoted by $[a_i\ ;\ b_i]$, with $i \in I$. Instead of $[a_i\ ;\ \infty)$, a persistent topological feature is denoted by $[a_i\ ;\ m)$ where $m = \max\{F\} + 1$. Note that the maximum persistent entropy corresponds to the situation in which all of the lines in the barcode are of equal length. Conversely, the value of the persistent entropy decreases as more lines of different lengths are present.

Consider again the persistent homology in Figure 5. Let the set $F = \{t_0, t_1, t_2, t_3\}$ be such that $t_0 = 0$, $t_1 = 1$, $t_2 = 2$ and $t_3 = 3$. Let $I = \{0, 1\}$ where 0 is the index of the unique line in `Dim 0` and 1 is the index of the unique line in `Dim 1` (both persistent in this case); then $m = 4$; the line 0 is $[0, 4)$; the line 1 is $[3, 4)$; yielding an entropy $H = 0.5$.

3.6. The Derived S[B] Model

At this point, enough information is available for defining a model of the complex system in the vision of the $S[B]$ paradigms, where global information constrains local interactions. Given an equilibrium condition (steady state) characterizing the system, an external or internal factor may alter this equilibrium and make the system react towards a new equilibrium [12]. If the system is observed at a certain moment, as when a sample of the data is taken, it may still satisfy the current equilibrium condition. If not, it has entered a critical transition. Depending on the future evolution, the system can go back to the previous steady state or reach a point in which it can not go back to the previous equilibrium. Thus, it starts an adaptation phase towards a new kind of known or unknown equilibrium.

In our methodology, persistent entropy is the global information that, at the S level, naturally constrains the local interactions at the B level, represented by the computational model discussed in Section 3.4. Note that we derive from the data and from the domain-specific knowledge a set of models, each of which gives an $S[B]$ representation of the system at the moment of the observation. The extraction of the local dynamics of the evolution of the system between two observations is beyond the scope of this work. Conversely, the global dynamics of the evolution of the system can be described in the form of a persistent entropy automaton.

3.7. Persistent Entropy Automaton

The evolution over time of the persistent entropy can be used for detecting when the system reacts to stimuli, *i.e.*, it starts a critical transition. Let us consider the chronogram plotting the values of the persistent entropy along the observed system data samples. Indeed, a peak in the chart means an abrupt change in the topology of the simplicial complexes. This information is used to identify the time interval in which the system possibly adapts towards a new equilibrium. On the other hand, a plateau in the chart corresponds to an equilibrium condition, namely a steady state. The relevant regions can then be used for deriving a state machine representing the global observed evolution of the system.

Finite state automata [28] are one of the most often used formalism for modeling systems (hardware, software, physical systems, and so on). They are a very familiar and useful model that extends the concept of the directed graph associating the nodes and the edges with the meaning of states and transitions, respectively. A state describes a static feature of the behavior of the system during its evolution, while a transition models how the system can evolve from one state to another. The basic formalism has been generalized in order to model timed [29] and hybrid systems [30]. In this paper, automata are equipped with topological information in order to model data-based complex systems.

We use the notion of persistent entropy as an observable, which is associated with the behavioral level B, to define the equilibrium conditions characterizing the steady states of the $S[B]$ system. For a more formal definition of the link between the observables and the dynamics of $S[B]$, we refer to [12]. The structural level S of the derived $S[B]$ model can be defined as a persistent entropy automaton (PEA), which is a tuple $(R, \Lambda, \rho_0, H, \rightarrow_S, L)$ where:

- R is a set of steady states;
- Λ is a set of names for the transitions;

- $\rho_0 \in R$ is the initial steady state;
- H is the observable variable, corresponding to the value of persistent entropy;
- $\rightarrow_S \subseteq R \times \Lambda \times R$ is a labeled transitions relation among steady states;
- $L(\rho)$ is a labeling function associating each state $\rho \in R$ with its equilibrium condition, depending on the values of H.

In a PEA, the notion of time is intrinsically present, which can be continuous or discrete. The value H of the persistent entropy must be defined for each instant t in the time domain. In our methodology, the time domain is discrete, and each discrete instant t corresponds to an observation of the data, *i.e.*, the value of the persistent entropy is the one calculated from persistent homology at each iteration and reported on the chronogram.

The dynamics of a PEA is described as follows. Whenever a PEA is in a steady state ρ, the current value of H (or a value derived from H) must satisfy the associated equilibrium condition $L(\rho)$. Time can elapse while the PEA stays in ρ, and the value of H changes accordingly. If at a certain point, the equilibrium condition $L(\rho)$ is no longer satisfied, the PEA must exit the state ρ and start executing a transition $\rho \xrightarrow{\lambda}_S \rho'$. This process is in general non-instantaneous; the PEA needs some time, *i.e.*, further changes in the value of H, in order to reach a point at which the equilibrium condition $L(\rho')$ of state ρ' is satisfied. At that moment, the PEA is again in a steady state, and the dynamics is again the one previously introduced.

4. M odel of the Case Study]S[B] Model of the Case Study

According to Jerne [13], the IN can be described by three main states: virgin, activation and memory. In the virgin state, there are no antibodies, but only non-specific T cells that start their activities after the activation signal sent by B cells that recognize the presence of pathogens. Then, the IN proliferates antibodies and reaches the activation state, which is not a steady state. After the activation, the IN performs the immunization, during which the antibodies play a dual role. In fact, an antibody can be seen both as a self-protein (a protein of the organism) or a non-self0protein (a pathogen to be suppressed). After the immunization, the IN reaches the memory state. This state represents a steady condition in which there is only a selection of antibodies. All of the transitions and states of IN are characterized by the fact that antibodies perform basically two actions: elicitation and suppression.

In the rest of this section, we report on the application of our methodology to the IN simulated with *C-ImmSim* [16,31]. We executed several (in the order of hundreds) simulations, each of them characterized by:

- a lifespan of 2190 discrete time ticks, where a tick corresponds to three days;
- a repertoire of at most 10^{12} antibodies, *i.e.*, the maximum number of antibodies available during the whole simulation;
- an antigen volume $V = 10$ µL.

During the simulation, an antigen is injected, and after an unknown (simulated) transient period of time, the same antigen is injected again.

From Data to Persistent Homology

In *C-ImmSim*, each idiotype (both antigens and antibodies) is represented with a bit-string, in our case of a 12-bit length. Two idiotypes A_i and A_j interact if and only if they are different, that is their Hamming distance $d(A_i, A_j)$ is such that $11 \leq d(A_i, A_j) \leq 12$. The pair-wise distances among all of the idiotypes are stored in an affinity matrix. However, this does not take into account the volume of the idiotypes. For this reason, in this case study, we decided to replace it with a coexistence matrix C where each element is a coexistent index. Given the Hamming distance $d(Ab_i(t), Ab_j(t))$ between two antibodies and their volumes $[Ab_i(t)], [Ab_j(t)]$ at time t, their coexistence index is defined as follows:

$$C_{Ab_{i,j}}(t) = \frac{d(Ab_i(t), Ab_j(t)) \cdot [Ab_i(t)] \cdot [Ab_j(t)]}{\sum_{l=1}^{n} [Ab_l(t)]} \tag{2}$$

where n is the number of antibodies.

Equation (2) expresses the fact that for lower values of affinity, the volume must be more significant, because the match between antibodies is less probable.

The coexistence matrix C is a symmetric matrix, and each element is taken to represent a weighted arc of an undirected graph. In this way, we obtained a graph representation of our data taken from the simulation. Figure 6 shows the weighted idiotypic network at the end of one simulation.

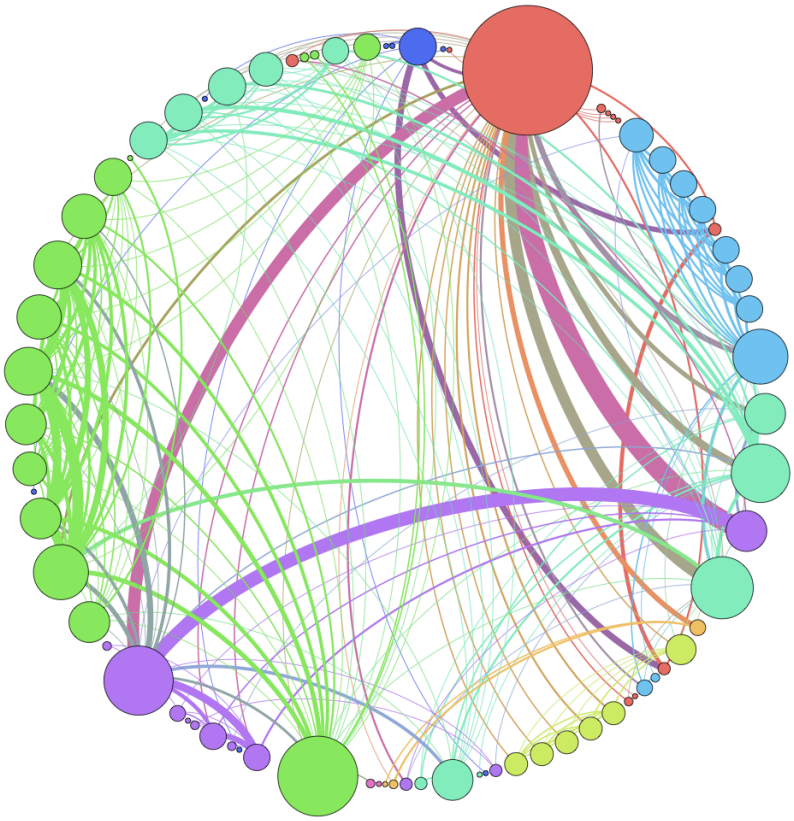


Figure 6. Example of the immune network at the end of a simulation. The thickness of the arcs is proportional to their weight; the diameter of the nodes is proportional to the number of incident edges.

The persistent homology of the filtered simplicial complex obtained from the weighted graph was computed with jHoles [24]. An example of the output is given in Table 1. For each data sample, we obtained a number of one-dimensional holes in the order of hundreds. We did not obtain any persistent n-dimensional hole with $n > 1$.

Table 1. Example of the output of jHoles with idiotypic network (IN) simulation data as the input. One connected component appeared at filtration value 0.0, which is persistent. The generator is the vertex called 16. Three 1-dimensional holes appeared at filtration values 7.0, 6.0 and 8.0, which are persistent. The four edges generating them are reported after the intervals.

β_0	$[0.0; \infty)$:	$\{[16]\}$
	$[7.0; \infty)$:	$\{[320, 3775], [256, 3775], [320, 3839], [256, 3839]\}$
β_1	$[6.0; \infty)$:	$\{[256, 3839], [256, 3711], [384, 3711], [384, 3839]\}$
	$[8.0; \infty)$:	$\{[260, 3835], [260, 3839], [256, 3835], [256, 3839]\}$

4.1. Idiotype Agent Behaviors as Higher Dimensional Automata

The persistent homology generated, as expected, a set of interaction matrices among entities. As introduced in Section 3.4, we now need to select a computational model to represent the behaviors of the agents (entities), and we need to use the domain-based knowledge in order to define their actions.

Among other computational models available for expressing concurrent computations, the true concurrent characteristic of the systems considered suggests the use of higher dimensional automata for modeling the behavior of the agents.

In 1991, Pratt published the formal definition of higher dimensional automata (HDA), based on the concept of n-dimensional objects [26]. HDAs generalize classical automata, allowing them to express non-interleaving concurrency. States and edges of a standard finite state automaton become n-dimensional objects where n is any finite dimension. The basic idea is that an n-dimensional object stands for an n-dimensional transition representing the concurrent execution of n actions.

Classical finite state automata are largely used for representing the behavior of concurrent and distributed systems. The components of these systems can be viewed as computational agents that execute their actions [32].

Figure 7 shows how to pass from a classical automaton to an HDA with a two-dimensional object, a surface. In the classical automaton on the left, the state at the bottom has two outgoing edges labeled a and b followed by b and a, respectively. Using an interleaving semantics, the automaton can perform both the two sequences of transitions labeled ab and ba. However, geometrically, the graph representing the automaton contains a hole, that is the interior of the square. Pratt suggested replacing the interleaving semantics with true concurrency by filling the hole (see Figure 7, right). The surface, depicted on the right, represents the simultaneous execution of a and b without imposing any order. Conversely, the "sculpting" of the inner surface by proper transformations allows one to represent with HDAs, also

classical non-determinism, *i.e.*, the presence of both possibilities of performing first a and then b or performing first b and then a.

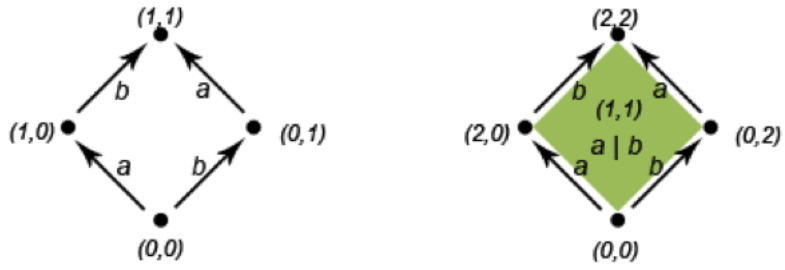
Figure 7. A classical finite state automaton representing the interleaving of actions a and b (**left**); the corresponding higher dimensional automaton (**right**).

Therefore, Figure 7 shows the behavior of two different computational agents running in parallel and performing the actions a and b, respectively. The classical automaton on the left corresponds to the implementation of the parallelism by interleaving, *i.e.*, only one action can be performed at time, so the agents must execute their actions in sequence, but both sequences are possible. The HDA on the right represents a true concurrent situation in which the two agents can effectively execute their actions at the same time.

A more intuitive and concise definition of HDAs was given in terms of Chu spaces [33,34]. A Chu space is a tuple $(A,\ r,\ X)$ over an alphabet Σ of statuses where A is a set of actions or events, X is a set of states and $r\colon (A \times X) \to \Sigma$ is a function representing the status of an action or event in a state. $\Sigma = \{0, 1\}$ is an appropriate alphabet for representing actions or events that may be either unstarted or finished, as in classical automata. Conversely, an extended alphabet $\Sigma = \{0, 1, 2\}$ is appropriate in the HDA setting, because actions or events can be unstarted, executing or finished, respectively.

Figure 8 shows the representation as Chu spaces of the automata in Figure 7. The function r of the Chu spaces is represented as a matrix. Every column represents a possible state of the automaton. For instance, on the left, state $s_0 = (0, 0)$ is one where both a and b are unstarted; state $s_1 = (0, 1)$ is one in which a is still unstarted and b is finished, and so on. On the right, state $s_1 = (0, 1)$ is the one in which action a is unstarted and action b is executing; state $s_5 = (1, 2)$ is the one in which action a is executing and action b is finished, and so on.

A graphical representation of a Chu space is often given using a Hasse diagram, which helps to easily deduce the set of admissible traces, dropping out automatically possibly forbidden states. In these diagrams, the computation proceeds in the upwards direction and represents an execution path in the automaton [33]. As an example, Figure 9 shows the Hasse diagram representing the Chu space of the HDA of Figure 7 (right).



$\Sigma=\{0,1\}$

r	s_0	s_1	s_2	s_3
a	0	0	1	1
b	0	1	0	1

$\Sigma=\{0,1,2\}$

r	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
a	0	0	0	1	1	1	2	2	2
b	0	1	2	0	1	2	0	1	2

Figure 8. On the left, a classical automaton together with the two-length alphabet of statuses and its Chu space representation as a matrix. On the right, a higher dimensional automaton (HDA) with its Chu space representation using the three-length alphabet.

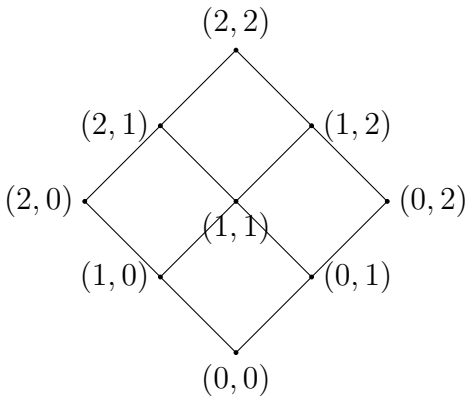


Figure 9. Hasse diagram representing the Chu space for the concurrent execution of two actions.

Applying Domain-Based Knowledge

From the Jerne model (see Section 2), it is well known that antibodies can perform elicitation and suppression actions simultaneously on different targets. Using this domain knowledge, we modeled the antibodies given by the generators of the persistent topological features as an HDA represented by a Chu space over the three-length alphabet.

We defined an action $a \in A$ of the Chu space as a triple $(action, source, target)$, where $action$ can be either elicit or reduce, and $source$ and $target$ are the identifiers of two antibody agents. Thus, a generic agent Ab_i can execute two possible actions concurrently, as described in the Chu space represented in Table 2. The corresponding Hasse diagram is the one already presented in Figure 9.

Table 2. Chu space representation of an HDA modeling a generic antibody Ab_i performing two actions on the same target Ab_j.

r	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$(elicit, Ab_i, Ab_j)$	0	0	0	1	1	1	2	2	2
$(reduce, Ab_i, Ab_j)$	0	1	2	0	1	2	0	1	2

Note that there is not a direct path between state $s_3 = (1, 0)$ to $s_1 = (0, 1)$, meaning that if action $(elicit, Ab_i, Ab_j)$ is executing and action $(reduce, Ab_i, Ab_j)$ is unstarted, then the first cannot go back to the unstarted status while the second becomes executing (or *vice versa*). Instead, from state $s_3 = (1, 0)$, it is possible to go to state $s_4 = (1, 1)$, meaning that both actions can be executing at the same time. Note that if in the Hasse diagram, the line between $s_3 = (1, 0)$ and $s_4 = (1, 1)$ were not present, that would have meant that action $(\text{reduce}, Ab_i, Ab_j)$ could not start executing before action $(\text{elicit}, Ab_i, Ab_j)$ became finished. Finally, note that the actions are bi-directional, *i.e.*, there is an elicit action from Ab_i to Ab_j and also an elicit action from Ab_j to Ab_i.

As a concrete example, we show the HDA modeling the behavior of two coupled antibodies Ab_1 and Ab_{13} resulting from the methodology applied to the case study. Table 3 shows the matrix of the Chu space representing such subsystem. Figure 10 shows the relative Hasse diagram.

Table 3. Chu space representing the subsystem $Ab_1 \mid Ab_{13}$.

r	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}
$(elicit, Ab_1, Ab_{13})$	1	0	1	2	0	0	0	0	0	0	1	1	1
$(reduce, Ab_1, Ab_{13})$	0	0	0	0	1	2	0	0	0	0	0	0	0
$(elicit, Ab_{13}, Ab_1)$	0	0	0	0	0	0	1	2	0	0	1	0	2
$(reduce, Ab_{13}, Ab_1)$	2	0	0	0	0	0	0	0	1	2	0	1	0
r	s_{14}	s_{15}	s_{16}	s_{17}	s_{18}	s_{19}	s_{20}	s_{21}	s_{22}	s_{23}	s_{24}	s_{25}	
$(elicit, Ab_1, Ab_{13})$	2	2	2	2	0	0	0	0	0	0	0	0	
$(reduce, Ab_1, Ab_{13})$	0	0	0	0	1	1	1	1	2	2	2	2	
$(elicit, Ab_{13}, Ab_1)$	1	0	2	0	1	0	2	0	1	0	2	0	
$(reduce, Ab_{13}, Ab_1)$	0	1	0	2	0	1	0	2	0	1	0	2	

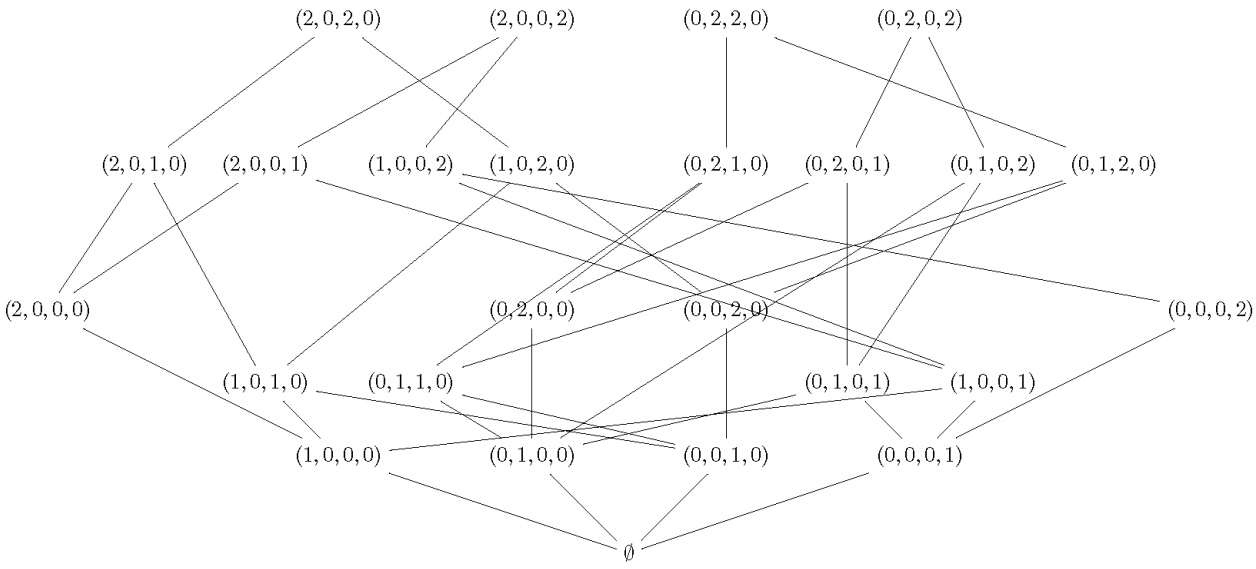


Figure 10. Hasse diagram for Chu space representing the subsystem $Ab_1 \mid Ab_{13}$.

4.2. Persistent Entropy Automaton for the Idiotypic Network

The value $H(t)$ of persistent entropy associated with every observation at time t was calculated accordingly to the definition in Section 3.5. Note that in this case, it is possible to build a chart of H over time, because we know that the data are subsequent, as they come from the same simulation. In general, for comparing two values of H, one should define the so-called mass function [35].

Figure 11 shows the chart of $H(t)$. Persistent entropy recognizes the dynamics of the immune system: a peak in the chart means immune activation, while a plateau represents the immune memory. An upward curve between a peak and a plateau means proliferation, while a backward curve means immune response. From the comparison of the charts, it is possible to recognize that the system is stimulated twice (Figure 11).

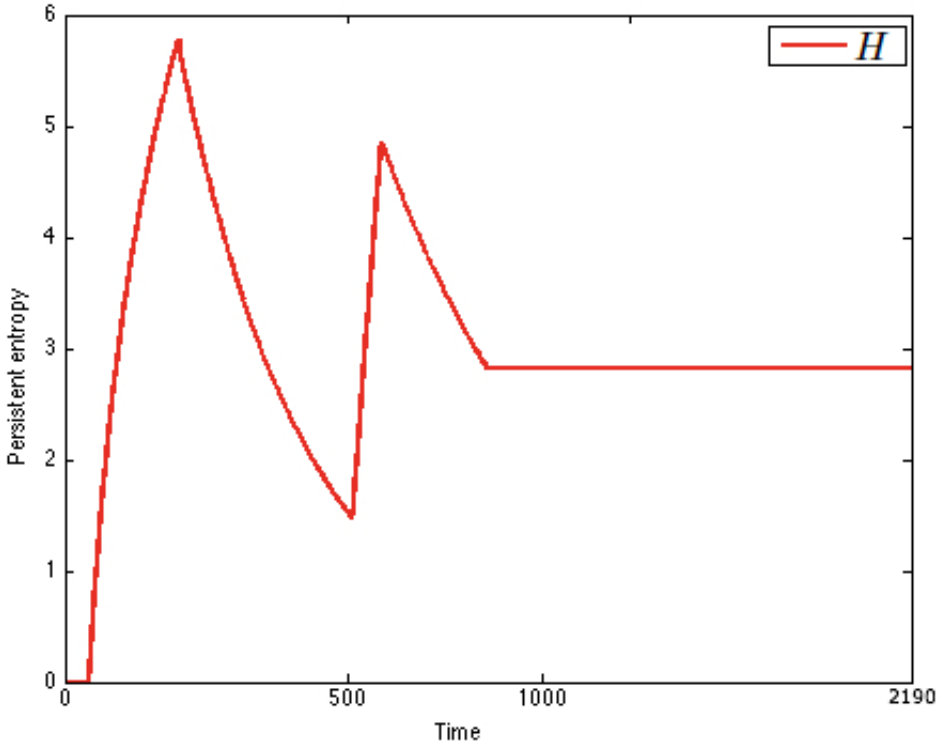


Figure 11. Persistent entropy of the immune system. The difference between the peaks amplitude is motivated by the fact that before the second peaks, the antibodies have been already stimulated, and the immune memory has been reached, so the system is more reactive and is faster in the suppression of the antigen; thus, the the entropy value for the steady state is $H = 2.87$.

Note that from the chart, it is also possible to derive a discrete estimation of the first and second derivatives of $H(t)$ by using finite differences. Analyzing the chart of persistent entropy, we were able to recognize two steady states, namely virgin and memory. The virgin state of IN is characterized by the equilibrium condition $H = 0 \wedge \dot{H} = 0$. This is the initial state in the PEA that we derived. The memory steady state corresponds to a plateau in the chart, and it is characterized by the equilibrium condition $H > 0 \wedge \dot{H} = 0$. This information is depicted in Figure 12 as the derived PEA.

Note that in the chart of Figure 11, two peaks are present. This reflects the fact that the system was stimulated twice and, thus, entered a critical transition followed by an adaptation phase from state memory to itself. This then suggests that a self-transition should be added to the state memory in the PEA.

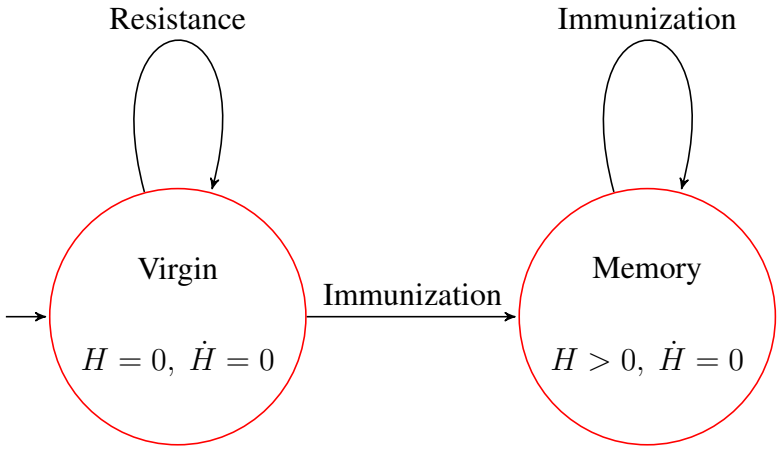


Figure 12. Persistent entropy automaton representing the S level of the $S[B]$ model of the idiotypic network.

Finally, even if we do not observe it in our simulation, we know from domain-specific knowledge that when in the virgin state, if the received stimulus is not grater than a certain threshold, then the immune activation does not start at all, and after a while, the system goes back to the initial state again. This means that also in the initial state of the PEA, a self-loop transition should be added.

5. Conclusions

In this work, we defined a new methodology, based on topological data analysis, for deriving a model for a complex system from a collection of data observations. Each observation is represented as a weighted graph, which is the scaffold of a corresponding topological object. The topological object is characterized by persistent homology, from which local and global information of the complex system is derived. The local information, about interacting computational agents, is represented by the new concept of topology-based interaction matrices. The global one is represented by persistent entropy, which is a recently-introduced entropy measure linking together information theory and topology. The $S[B]$ paradigm is the framework in which the methodology extracts and defines the models, yielding a new kind of entangled model. The global dynamics through all of the observations is described by the newly-introduced notion of persistent entropy automaton. We applied the methodology for modeling the idiotypic network of the mammalian immune system with data taken from the simulator *C-ImmSim*.

We showed that the methodology can be effectively used to model a complex system, towards a new topological field theory of data [36].

Acknowledgments

The authors thank Mario Rasetti for being their mentor, unique and invaluable during the years of the Topology driven methods for complex systems (TOPDRIM) project and especially for his helpful suggestions, but also criticisms. We acknowledge the financial support of the Future and Emerging Technologies (FET) program within the Seventh Framework Programme (FP7) for Research of the European Commission, under the FP7 FET-Proactive Call 8 Dynamics Multi-level Complex Systems

(DyMCS), Grant Agreement TOPDRIM, Number FP7-ICT-318121. Peter Sloot also acknowledges partial support from the Russian Scientific Foundation, Proposal #14-21-0137.

Author Contributions

Emanuela Merelli and Matteo Rucco conceived the methodology; Matteo Rucco designed and performed the data analysis; Peter Sloot supervised the application of the methodology on the case study; Luca Tesei and Emanuela Merelli wrote the paper.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

1. Stein, D.; Newman, C. Nature versus nurture in complex and not-so-complex systems. In *ISCS 2013: Interdisciplinary Symposium on Complex Systems*; Springer: Berlin Heidelberg, Germany, 2014; pp. 57–63.
2. Boccara, N. *Modeling Complex Systems*; Springer-Verlag: New York, USA , 2010; Second edition.
3. Jennings, N.R. An agent-based approach for building complex software systems. *Commun. ACM* **2001**, *44*, 35–41.
4. Carlsson, G. Topology and data. *Bull. Am. Math. Soc.* **2009**, *46*, 255–308.
5. Zomorodian, A.J. Topology for computing. In *Cambridge Monographs on Applied and Computational Mathematics*; Cambridge University Press: Cambridge, UK, 2009; Volume 16.
6. Ibekwe, A.M.; Ma, J.; Crowley, D.E.; Yang, C.H.; Johnson, A.M.; Petrossian, T.C.; Lum, P.Y. Topological data analysis of escherichia coli o157: H7 and non-o157 survival in soils. *Front. Cell. Infect. Microbiol.* **2014**, *4*, 122.
7. De Silva, V.; Ghrist, R. Coverage in sensor networks via persistent homology. *Algebraic Geom. Topol.* **2007**, *7*, 24.
8. Chan, J.M.; Carlsson, G.; Rabadan, R. Topology of viral evolution. *Proc. Natl. Acad. Sci. USA* **2003**, *110*, 18566–18571.
9. Petri, G.; Expert, P.; Turkheimer, F.; Carhart-Harris, R.; Nutt, D.; Hellyer, P.J.; Vaccarino, F. Homological scaffolds of brain functional networks. *J. R. Soc. Interf.* **2014**, *11*.
10. Edelsbrunner, H.; Harer, J. *Computational Topology: An Introduction*; American Mathematical Soc.: Providence, USA, 2010.
11. Merelli, E.; Pettini, M.; Rasetti, M. Topology driven modeling: The IS metaphor. *Nat. Comput.* **2015**, *14*, 421–430.
12. Merelli, E.; Paoletti, N.; Tesei, L. Adaptability Checking in Complex Systems. *Sci. Comput. Program.* **In Press**, doi:10.1016/j.scico.2015.03.004.
13. Jerne, N.K. Towards a network theory of the immune system. *Ann. Immunol.* **1974**, *125*, 373–389.
14. Holland, J.H. Complex adaptive systems. *Daedalus* **1992**, *121*, 17–30.
15. Parisi, G. A simple model for the immune network. *Proc. Natl. Acad. Sci. USA* **1990**, *87*, 429–433.

16. Bernaschi, M.; Castiglione, F. Design and implementation of an immune system simulator. *Comput. Biol. Med.* **2001**, *31*, 303–331.
17. Hoffmann, G. A theory of regulation and self-nonself discrimination in an immune network. *Eur. J. Immunol.* **1975**, *5*, 638–647.
18. Mikk, E.; Lakhnech, Y.; Siegel, M. Hierarchical Automata as Model for Statecharts. In *Advances in Computing Science (ASIAN'97)*; No. 1345 in Lecture Notes in Computer Science; Springer: Berlin Heidelberg, Germany, 1997; pp. 181–106.
19. Jonsson, J. *Simplicial Complexes of Graphs*; Springer-Verlag: Berlin Heidelberg, Germany, 2008.
20. Bron, C.; Kerbosch, J. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM* **1973**, *16*, 575–577.
21. Bandelt, H.J.; Chepoi, V. Metric graph theory and geometry: A survey. *Contemp. Math.* **2008**, *453*, 49–86.
22. Adams, H.; Tausz, A.; Vejdemo-Johansson, M. javaPlex: A research software package for persistent (co) homology. In *Mathematical Software-ICMS 2014*; Springer: Berlin Heidelberg, Germany, 2014; pp. 129–136.
23. Petri, G.; Scolamiero, M.; Donato, I.; Vaccarino, F. Topological strata of weighted complex networks. *PLoS ONE* **2003**, *8*, e66506.
24. Binchi, J.; Merelli, E.; Rucco, M.; Petri, G.; Vaccarino, F. jholes: A tool for understanding biological complex networks via clique weight rank persistent homology. *Electron. Notes Theor. Comput. Sci.* **2014**, *306*, 5–18.
25. Merelli, E.; Rasetti, M. Non locality, topology, formal languages: New global tools to handle large data sets. *Proced. Comput. Sci.* **2013**, *18*, 90–99.
26. Pratt, V. Modeling concurrency with geometry. In Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Orlando, FL, USA, 21–23 January 1991; pp. 311–322.
27. Rucco, M.; Castiglione, F.; Merelli, E.; Pettini, M. Characterisation of the idiotypic immune network through persistent entropy. In *Proceedings of ECCS 2014 European Congerence on Complex Systsems*; To appear in Springer Proceedings in Complexity; Springer International Publishing Switzerland: Cham, Switzerland, In Press.
28. Hopcroft, J.E.; Motwani, R.; Ullman, J.D. Introduction to automata theory, languages, and computation. *ACM SIGACT News* **2001**, *32*, 60–65.
29. Bernardo, M.; Tesei, L. Encoding timed models as uniform labeled transition systems. In *Computer Performance Engineering*; Springer: Berlin Heidelberg, Germany, 2013; pp. 104–118.
30. Henzinger, T.A. *The Theory of Hybrid Automata*; Springer: Berlin Heidelberg, Germany, 2000.
31. Mancini, E.; Castiglione, F.; Bernaschi, M.; de Luca, A.; Sloot, P. HIV reservoirs and immune surveillance evasion cause the failure of structured treatment interruptions: A computational study. *PLoS ONE* **2012**, *4*, e36108.
32. Milner, R. *Communication and Concurrency*; Prentice-Hall, Inc.: Upper Saddle River, USA, 1989.
33. Gupta, V. *CHU Spaces: A Model of Concurrency*. Ph.D. Thesis, Stanford University, Palo Alto, CA, USA, 1994.
34. Pratt, V. Higher dimensional automata revisited. *Math. Struct. Comput. Sci.* **2000**, *10*, 525–548.

35. Stewart, W.J. *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*; Princeton University Press: Princeton, USA, 2009.
36. Rasetti, M.; Merelli, E. The topological field theory of data: A program towards a novel strategy for data mining through data language. *J. Phys. Conf. Ser.* **2015**, 626 012005.