A methodology to evaluate the evolution of networks using topological data analysis

Joao Pita Costa
University of Rijeka, Croatia
Quintelligence, Slovenia

Tihana Galinac Grbac
University of Rijeka, Croatia

ABSTRACT

Networks are important representations in computer science to communicate structural aspects of a given system of interacting components. The evolution of a network has several topological properties that can provide us information on the network itself. In this paper, we present a methodology to compare the the topological characteristics of the evolution of a network, encoded into a (persistence) diagram that tracks the lifetimes of those features. This will enable us to classify the evolution of networks based on the distance between the diagrams that represent such network evolution. In that, we also consider complex vectors that bring a complementary perspective to the distance-based classification that is closer to the computational methods, aims to enhance the computational efficiency of those comparisons, and that is by itself a source of open research questions.

Categories and Subject Descriptors

G.2.2 [**Mathematics of Computing**]: Discrete Mathematics — applications

General Terms

Algorithms, Measurement, Reliability, Experimentation, Theory.

Keywords

Network, undirected graph, persistent homology, computational topology, persistence diagram.

1. INTRODUCTION

1.1 Comparing the topology of the evolution of networks

Networks that change as a function of time - known as evolving networks - are a natural extensions of undirected graphs (i.e., standard (static) networks). Almost all real world networks evolve over time, either by adding or removing nodes or edges. The example of scientific collaboration analysis, such as in the example of Figure 1 shows such a network. The analysis of the evolution of a network is a matter of interest transversal to many fields of knowledge, from social network analysis and scientific collaboration to computational biology. A standard example is the network dynamics of a social network such as Twitter should consider an evolution through time where new nodes come up as new members join, and new edges are created mirroring the new relationships between members that appear [1]. Often all of these processes occur simultaneously in social networks. Collaborative networks are a prime example of evolving networks, where nodes represent authors and edges represent scientific collaborations. This is illustrated in Figure 1. It shows the plot of three phases of
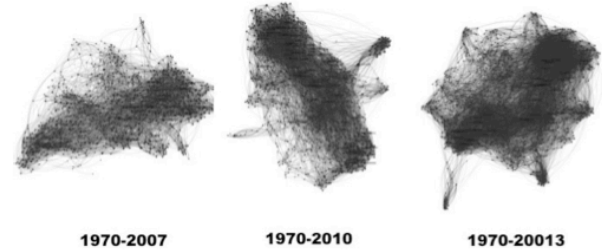
an instance in the scientific community in Slovenia

Fig. 1. The evolving network ScienceAtlas of the collaborations in scientific works by Slovenian researchers, evolving over a 9-year period with 3-year leaps. Each node represents an author and each edge represents a collaboration. The nodes with degrees smaller than 20 are filtered out so that the networks are not too large to be visualized.
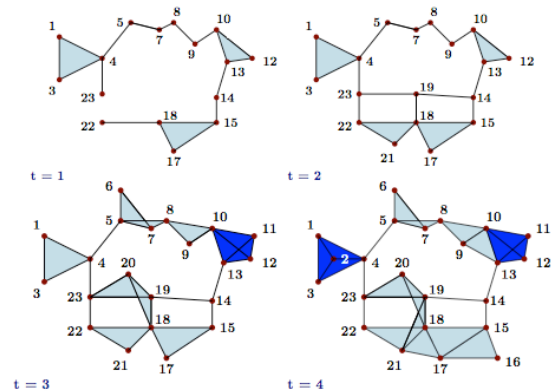

Fig. 2. An example of an evolving network based on an undirected graph that is growing over time by adding new edges and nodes at each step.

[14] using ScienceAtlas [13], a web portal available at scienceatlas.ijs.si integrating data about 35272 researchers, 5716 projects, 82905 publications and 17190 video lectures. This too allows visualizing collaboration and competences of the researchers [15]. A biological network, on the other hand, is an approximate mathematical representation of connections found in ecological, evolutionary, and physiological research, among others. An example of a relevant application of such analysis of biological networks with respect to human diseases is network medicine. It considers networks in biological systems containing many components connected within complicated relationships but organized by simple principles [1]. In this paper, we focus on the comparison of the evolution of two (or more) given networks. Our approach considers topological data analysis (TDA), allowing us to encode the topological features of the corresponding evolving networks onto diagrams, and using standard methods to compute
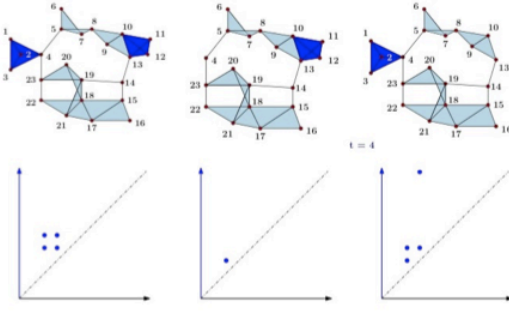
Fig. 3. The presented methodology applied to the comparison of the evolution of three networks originated from the evolving network represented in Figure 2. Each evolving network is associated with one persistence diagram that encodes its topology. This permits us to visualize the relevant topological features of the evolution of the network, and to compute distances between the evolution of two or more networks.
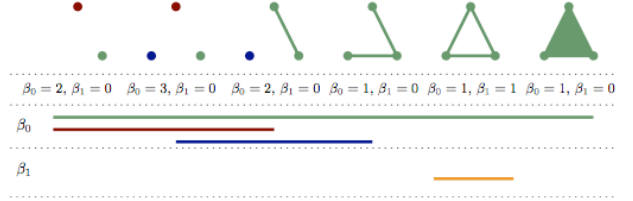

Fig. 4. The computation of persistent homology on a simplicial complex changing in time [3]. The colors correspond to the topological features to which the lifetime is tracked in the persistence barcode below. The Betti numbers indicate the number of connected components β_0 in dimension zero, the number of holes in the network β_1 in dimension one, and the number of tunels and voids β_2 in dimension two.

distances between them. In that, we can classify networks according to the distance between the topology of their evolution. The TDA approach to the study of networks is not itself new. It had several widespread applications from collaboration networks [3] to functional brain networks [16]. There are several ways of considering a height function in a network including: (i) considering weights in the edges of the network - *weighted network* - and then having the function built by threshold those weights [18]; (ii) measuring the distance from each node to each other by counting the minimal number of edges between them and then building the height function based on that distance [11]; among others. This permits us to use persistent homology over such height function. Another possibility is to consider the maximal cliques as the simplicial complexes (named *clique complexes) that* feed the persistence algorithm and proceed with the computation directly over that [5]. We used the latter approach to compute the persistence of the networks generated for the purpose of this paper.

1.2 Basic notions in persistent homology

Topology is a field of study in mathematics concerned in the quality aspects of an object. It focus on the properties that are preserved through deformations, twistings, and stretchings of the given continuous objects (e.g. linear maps) in multidimensional scenarios. Computational topology takes advantage of simplification methods (e.g. the triangulation of a space) to permit the computation of topological invariants. One of those computations is homology, which evaluates the connectedness of, e.g., a network at different dimensions separately. Thus, homology is a natural choice when it comes to the study of the topology of a network. Now, if we consider a monotone function describing the time variable in, e.g., an evolving network, we can track its homology changes. This notion is known as persistent homology and is rooted in TDA, allowing for retrieving the essential topological features of an object [2]. Formally, persistent homology computes the topological features of a growing sequence of spaces $\varnothing = X_0 \subseteq X_1 ... \subseteq X_n = X$, known as a *filtration* of the space X. $H_i(X)$ is the i-th homology group of X, with an associated i-th Betti number of X, βi, corresponding to the measure of connectedness in the i-th dimension (cf. [5]). Using the inclusion maps X_j in X_{j+1} we can identify copies of Z_2 in the homology groups $H_i(X_j)$ and $H_i(X_{j+1})$ of a filtration and track where the homology changes. We do that by recording when a new copy appears (i.e. "is born"), and when an existing copy

persists or merges to an existing one (i.e. "dies"). That persistence of the topological feature is tracked by a lifetime bar (as shown in Figure 4) that can be equivalently represented by an ordered pair *(x,y)*, where x is the birth time and y is the death time. The multiset of all such points exists in the plane subset defined by $0 < x < y$ that encodes the topology of a space and is known as *persistence diagram*. Several topological features can have the same lifetimes and therefore some of the points in the persistence diagram are repeated in the multiset. We refer to their amount as *multiplicity*. We consider the infinite points in the diagonal as points of the persistence diagram with null lifetime. The standard method to compare two persistence diagrams – called *bottleneck distance* - measures the cost of finding a correspondence between their points. It identifies the closest matching elements of each persistence diagram and determines the global distance based on what is the biggest of those distances. The cost of taking a point $p = (p_1, p_2)$ to a point $q =(q_1, q_2)$ in R^2 is given by the L_∞ norm $| p - q |_\infty = max\{|p_1-q_1| , |p_2-q_2|\}$. Then, the bottleneck distance between persistence diagrams X and Y is computed by taking the infimum over all such matchings, i.e., $d_B(X,Y)=\inf_\eta \sup_{x \in X} \|x - \eta(x)\|_\infty$, where the inf. is taken over all bijections η from X to Y. Each point with multiplicity k in a multiset is interpreted as k individual points, and the bijection is interpreted between the resulting sets [4].

1.3 The motivation of EVOSOFT

Nowadays, software systems start to interconnect to provide new and innovative applications and services that drive new development opportunities in all domains. Therefore these software systems have gradually evolved into large-scale complex systems and we lack models for their further management and evolution. One of the key aspects of such systems is the ability to model and predict their behavior to achieve the required quality of operations to fulfill human expectations in all domains. In that, the project Evolving Software Systems: Analysis and Innovative Approaches for Smart Management (EVOSOFT) aims to understand how abstract software structures can be used to model global system properties (e.g. fault distributions). Understanding how to use software structure to model fault distributions can help us to improve system reliability. EVOSOFT observes software structure as networks with nodes representing various software functions that are interconnected to each other by function calls. In particular, a software graph structure considers nodes as program functions (e.g. classes in object oriented paradigm, functions or modules in functional programming) and edges as function calls or signals transferred in communication among these program functions. EVOSOFT aims to observe how large software systems evolve from version to version, and understand the relationship between the change in software structure during

Fig. 5. The methodology diagram to encode a compare the topology of the evolution of two (or more) networks.

its evolution, and the change in software fault distributions across its structure. Previous empirical studies in [9], [10] and [18] show that communication structures among the program functions significantly influence system fault distributions. This is what motivated us to further explore this relationship..

2. USE-CASE METHODOLOGY TO ENCODE AND COMPARE EVOLVING NETWORKS

The problem of tracking and comparing the evolution of networks can be very demanding and complex due to the combinatorial properties of networks. In the following section we shall describe the methodology diagram to encode a compare the topology of the evolution of networks (as illustrated in Figure 5). It considers persistent homology to encode the topological features of the evolution of a network using persistence diagrams. In that, we first provide the evolving network given by one Boolean adjacency matrix for each phase of network development. We then compute the persistent homology of the evolving network by feeding the concatenated matrices a suitable algorithm. It will encode the topology of each evolving network, representing it by one unique persistence diagram each. Finally, we measure the bottleneck distance between persistence diagrams to identify how close are the evolving networks to each other based on their topology. To the purpose of this paper, we used the software library *Perseus* [16] to compute the homology of a the evolving network represented in Figure 3, given by the graph's Boolean adjacency matrix. The network is provided to *Perseus* as a list of cliques including the time of appearance. The output of that procedure is a persistence diagram that corresponds to the topological changes within the evolution of that network. The evolving network *A* on the left has four stages as illustrated in Figure 3. The evolving networks *B* and *C* are variations of the evolution of the end network in *A* with different phases at time *t=2*, as represented in 2. To compare the evolution of networks we consider the distance between the corresponding persistence diagrams, using the bottleneck distance. This permits a fast computation of the distance between the (persistence diagrams representing the) topology of two evolving networks. In the case of the persistence diagrams encoding the topology of evolving networks A, B and C represented in Figure 2, we get $d(A,B)=0$ and $d(B,C)=d(A,C)=1$. This discards the points with infinite persistence that are less relevant when considering dimension 1 diagrams. The computations were done using the TDA package available in R [7]. In this example we can explore the distance between several possible evolutions of a network. In it, shows how TDA can contribute to better understand the behavior of a certain network.

3. THE EVOSOFT EXPERIMENTS

For the purpose of this research we will use the EVOSOFT motivation to generate networks that fit that scenario and allow us to compare the evolution of networks in that context. In these preliminary experiments we shall consider data representing the evolution of networks based on the empirical analysis of the

evolution of complex software systems [8]. In these experiment we will generate networks with labeled nodes - not ordered pairs in R^2 - and extract all maximal cliques from it. The maximal cliques serve us to construct clique complexes with which we are able to later on compute the topology of those networks. In these experiments we shall obtain the EVOSOFT evolving networks provided by their graph's Boolean adjacency matrix. Those matrices must be consistent with the evolution of the network in the sense that existing maximal cliques in phase *i* must maintain or enlarge in the phase *i+1* during the updates of a software version. The persistence diagrams computed by *Perseus* shall exhibit the encoded topology of evolving networks corresponding to different pieces of software. The comparison between the topology of a pair of evolving networks given by the adjacency matrix is given by the bottleneck distance between the corresponding diagrams. That distance can be computed using the R library [7]. When considering other evolving networks we can calculate the pairwise distance between all of them and consider single linkage clustering based on this metric (as in earlier TDA applications to gene expression data as in [12]) to allow classification based on the topology of network evolution.

4. COMPARISON THROUGH COMPLEX VECTORS

A possible algebraic representation of persistence diagrams is offered by complex polynomials. The method layed out in [6] can lead to avoid tedious and less meaningful computations of bottleneck distance, since far polynomials represent far persistence diagrams (the converse is known not to be true). A fast comparison of the coefficient vectors can reduce the size of the database to be classified by the bottleneck distance. We can then focus on close persistence diagrams for which we want to calculate precise measures. This should complement existing methods, rising the efficiency of computations for large evolving networks. Given a persistence diagram D described by its points $p_1 = (u_1,\ v_1),\ \dots\ ,\ p\_s = (u_s,\ v_s)$ with multiplicities $r_1,...,r_s$, respectively, the method considers complex numbers $z_1=u_1+iv_1,..., z_s = u_s+iv_s$. This allows us to associate to D the complex polynomial $f_D(t) = Pi^s_{j=1}(t - z\_j)^{rj}$ where r_j is the multiplicity of the point p_j. It was shown in [5] that the first k coefficients are the ones carrying most of the relevant information and, therefore, the choice of a threshold k can reduce the computational complexity. The unpublished 2-part algorithm by the authors of [6] permits us to input a persistence diagram in order to compute a complex vector out of it. Then the same algorithm compares two complex vectors corresponding to two persistence diagrams to output a float corresponding to the distance between those vectors. At the moment, this approach to convert persistence diagrams into complex vectors can be applied only when neglecting points with infinite persistence. In the running example we get the polynomial $p_A=(t-1-3i)(t-2-4i)(t-3-4i)=p_B$ and $p_C=(t-1-3i)(t-2-4i)^2$, not considering points of infinite persistence. We then develop the polynomials to identify their coefficients into a complex vector. The distance between the three complex vectors corresponds to a basic classification of the given evolving networks. This is not a dense case where we would need additional tools like complex vectors. Though, real life examples of evolving networks are appropriate cases of such needs due to their inherent complexity..

5. CONCLUSIONS AND FURTHER WORK

In this paper we have discussed the topological data analysis of evolving networks. In that we presented a method to encode the

topology of the evolution of a given network through a persistence diagram, and its potential for a classification based on a chosen distance between diagrams. The inherent complexity of an evolving network demands for the data simplification methods to be available and appropriate to the nature of the considered object. In that, the TDA-based methodology in this paper can contribute to the analysis and interpretation of evolving networks and their behaviour. The experiments in real data are valuable to improve this method. In that, the collaborations with the earlier mentioned *Slovenian Science Atlas* would be welcome, allowing us to further explore the interpretation of the topology of the evolution of these collaborative networks and the distance between them. Further work includes the processing of EVOSOFT existing networks, as well as the interpretation of results in the context of that field of knowledge. It can provide new challenges specific to the available data and to its role and usage in the field. In particular, the interpretation of the persistent topological features captured in EVOSOFT experiments represents a relevant open problem that requires a deeper analysis based on the EVOSOFT expertise and the manipulation of the topological results. Lastly, the mathematical development of the complex vector method, that contributes to the study of evolving networks in general, is a rather computational method that is suitable to the application of compatible algorithms, allowing potential engineering applications. Moreover, it is itself a great source of open mathematical problems that we shall consider in further research (e.g. *stability* [6]).

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] A. L. Barabasi. Network medicine from obesity to the diseasome. *New England Journal of Medicine*, 357(4):404–407, 2007.

[2] G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[3] C. J. Carstens and K. J. Horadam. Persistent homology of collaboration networks. *Mathematical problems in engineering*, 2013.

[4] H. Edelsbrunner and J. Harer. Persistent homology - a survey. *Contemporary mathematics*, 453:257–282, 2008.

[5] H. Edelsbrunner and J. L. Harer. Computational Topology. *American Mathematical Society*, Providence, RI, 2010.

[6] B. D. Fabio and M. Ferri. Comparing persistence diagrams through complex vectors. In *International Conference on Image Analysis and Processing*, 294–305, 2015.

[7] B. T. Fasy, J. Kim, F. Lecci, C. Maria, and V. Rouvreau. TDA package for R. *cran.r-project.org/web/packages/TDA*. Accessed: 2017-09-05.

[8] T. Galinac and S. Golubić. Project overlapping and its influence on the product quality. In *Proceedings of the 8th International Conference on Telecommunications*, 2005. ConTEL 2005, 2:655–662, 2005.

[9] T. G. Grbac and D. Huljenic. On the probability distribution of faults in complex software systems. *Information & Software Technology*, 58:250–258, 2015.

[10] T. G. Grbac, P. Runeson, and D. Huljenic. A second replicated quantitative analysis of fault distributions in complex software systems. *IEEE Trans. Software Eng.*, 39(4):462–476, 2013.

[11] D. Horak, S. Maletić, and M. Rajković. Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(3):P03034, 2005.

[12] M. Juda. Topological structures in gene expression data. unpublished work presented at the *Genetic Analysis Workshop 19*, 2014.

[13] M. Karlovčec. Scienceatlas. *http://scienceatlas.ijs.si/*. Accessed: 2017- 09-05.

[14] M. Karlovčec, B. Lužar, and D. Mladenić. Core-periphery dynamics in collaboration networks: the case study of slovenia. *Scientometrics*, 109.3:1561–1578, 2016.

[15] M. Karlovčec, D. Mladenić, M. Grobelnik, and M. Jermol. Conceptualization of science using collaboration and competences. *The Electronic Library*, 34.1:2–23, 2016.

[16] H. Lee, M. K. Chung, H. Kang, B.-N. Kim, and D. S. Lee. *Discriminative persistent homology of brain networks*. IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 841– 844, 2011.

[17] V. Nanda. Perseus, the persistent homology software. *www.sas.upenn.edu/~vnanda/perseus*. Accessed: 2017-9-5.

[18] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Topological strata of weighted complex networks. *PloS one*, 8(6):e66506, 2013.

[19] J. Petrić and T. G. Grbac. Software structure evolution and relation to system defectiveness. In *18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14*, London, England, United Kingdom, May 13-14, 2014, 34:1–34:10, 2014.