System Network Complexity: Network Evolution Subgraphs of System State Series

Article *in* IEEE Transactions on Emerging Topics in Computational Intelligence · October 2018

DOI: 10.1109/TETCI.2018.2848293

Animesh Chaturvedi

Indian Institute of Information Technology, Design and Manufacturing Jabalpur

17 PUBLICATIONS **112** CITATIONS

System Network Complexity: Network Evolution Subgraphs of System State series

Animesh Chaturvedi and Aruna Tiwari

***Abstract*— Era of computation intelligence leads to various kinds of systems that evolves. Usually evolving system contains evolving inter-connected entities (or components) that makes evolving networks for the system state series $\mathbb{SS}$ = {S_1, S_2… S_N} created over time. Where, S_i represents i^{th} system state. In this paper, we introduce an approach for mining *network evolution subgraphs* such as *network evolution motifs* (NEMs) and *network evolution graphlets* (NEGs) from a set of evolving networks. The NEGs information are then used to calculate the *System State Complexities* (SSCs) and *Evolving System Complexity* (ESC). The *System State Complexity* (SSC) is the complexity of single system state. Extensively, the ESC is time-varying complexity of a state series aggregated over time. We proposed an algorithm named *System Network Complexity* (SNC) that uses three algorithms for mining NEGs, SSCs, and ESC. The SNC analyses a pre-evolved state series of an evolving system. We prototyped the technique as a tool named as *SNC-Tool*, which is applied on four different domains including: software system, natural language system, retail market basket system, and IMDb movie genres system. This is demonstrated as experimentation reports containing retrieved NEGs, NEMs, SSCs, and ESC for six real-world evolving systems collected from open-internet repositories.**

***Index Terms*— Data mining, Systems engineering and theory, Network theory (graphs), Complexity theory.**

I. Introduction

An evolving system maintains changeability in the form of multiple states [1][2][3]. A repository stores and manages system state series (like a data series [4]) at centralize database. It is challenging to mine evolution information in multiple states of an evolving system. Change analysis of various states can provide an overview of evolution happened in the evolving system. Analysis of such evolving systems can be done using techniques like non-stationary learning [5], change mining [6], evolution mining [7], and incremental mining [8].

We assume that a system state can be represented using a network of relationship(s) between inter-connected entities. A network that changes over time are referred as a *time-varying networks* [9], *dynamic networks* [10], or *temporal networks* [11], or *multilayer networks* [12], or *evolving networks* [13].

Animesh Chaturvedi is with the Indian Institute of Technology Indore, Indore, MP, India. Email: animesh.chaturvedi88@gmail.com.
Aruna Tiwari is with the Indian Institute of Technology Indore, Indore, MP, India. E-mail: artiwari@iiti.ac.in.

An *evolving network* is a finite sequence of directed graphs, which describes the state S_i at i^{th} time instance. This makes series of evolving networks {EN_1, EN_2... EN_N}, where an EN_i represents a state S_i. An evolving network $EN_i = (V, E)$ has a set of vertices V and a set of edges E. The graph $H = (V', E')$ is a subgraph of EN_i if $V' \subseteq V$ and $E' \subseteq E$. The frequency of subgraphs H is the number of occurrence of H in EN_i.

Subgraphs are the building blocks that construct its network; thus, subgraphs are used to study network properties. One such property is *frequent subgraphs*. If the frequency of H is higher than a threshold value, then H is a *frequent subgraph* of EN_i. Few frequent-subgraph mining techniques are AGM [14], FSG [15], and gSpan [16]. A *network motif* [17][18] is a well-known kind of frequent subgraph that are statistically obtained using random graphs. The *graphlet* [19][20] is another well-known kind of subgraph, which is a small connected induced subgraph.

A graph can be useful to calculate system complexity as a measure of interaction between various system entities as nodes. To calculate system network complexities, we use the well-known and widely appreciated metric, *cyclomatic complexity* by McCabe's [24]. It is a metric to calculate the complexity from a graph of system (especially software). The cyclomatic complexity is a graph metric that is proven useful for determining the complexity of computer programs when applied to their control and decision flow graphs. Mathematically, cyclomatic complexity is a number of regions or a number of linearly independent paths in the graph and calculated using the formula, $C = (e - n + 2)$, where 'e' stands for the number of edges and 'n' stands for the number of nodes in the graph. In this paper, we demonstrate that "cyclomatic complexity is also applicable to systems that are represented as a network (or graph)". Such result is non-obvious and non-trivial information about an evolving system.

Suppose a given system *state series* $\mathbb{SS}$ is represented as a set of evolving networks of evolving inter-connected entities. We are motivated to capture the *graphlets information* of a state and use it to calculate *system state complexity*. Additionally, we aggregated graphlets information of all states over the time to retrieve *network evolution graphlets information* and use it to calculate *evolving system complexity*. Rest of the paper is organized as following.

Section II describes network subgraph mining for a state. Section III presents our proposed keyword definitions. Section IV contributes an approach *System Network Complexity* (SNC) to analyze an evolving system. Section V describes SNC-Tool based on proposed work. Section VI demonstrates the application of SNC-Tool on six evolving systems of four different domains. Lastly, in Section VII we discuss related contributions, and in Section VIII we conclude the paper.

II. Preliminaries for Network Subgraph Mining

This section presents network subgraph mining for a system state. For better understanding of the paper, we will introduce some auxiliary notation and search space for subgraph mining.

Initially, choose a *subgraph size* to fix the number of vertices in a subgraph. We discriminated various subgraphs (or motifs or graphlets) by enumerating subgraphs from 0 to M. The G_j denotes jth graphlet. The $\{G_0, G_1, G_2 \ldots G_M\}$ denotes a set of all possible graphlets for a given *graphlet size*, where M is the total possible graphlets. *Graphlet frequency* is the count of induced subgraphs in a network and denoted as $freq_j$ corresponding to the G_j. The $\{freq_0, freq_1, freq_2 \ldots freq_M\}$ denotes a set of *graphlet frequencies* for all possible graphlets.

These graphlets can be retrieved with a network subgraph mining technique. Search space of subgraph mining in our case is a system network containing connections between entities. Network subgraph mining retrieves induced subgraph information i.e. *graphlets information*, which is a doubleton set of graphlet and its frequency in a given network. We denoted the retrieved *graphlets information* as

$$graphlets\_info = < G_j, freq_j > \mid 0 \le m \le M \qquad \text{-- (1)}$$

where, G_j is a graphlet occurring at least $freq_j$ number of times in the network, and j is in enumeration of graphlet. Such that m is the number of retrieved graphlets for state S_i.

Different pre-processing techniques make different networks (i.e. relationship between entities as source and target nodes). The pre-processing depends upon the kind of the evolving system, i.e. different system has different pre-processing to make a series of evolving networks. This means a system pre-processing requires system domain expert, who can identify appropriate entities and relationship between those entities. The selection of relationship between inter-connected entities depends upon the required result (as final output). Even a system itself has different kinds of complex pre-processing to generate its evolving network. For example, while pre-processing: for a software system, procedure-calls are entity-connections relationships; and for a natural language system, word-sequences are entity-connections relationships.

Network subgraph mining can solve crucial problems by retrieving insight about characteristics and internal working of system network. In Figure 1, a state S_i is pre-processed to make an evolving network EN_i using intrinsic property for inter-connected entities of the evolving system. Further, the subgraph mining tool takes EN_i as input to retrieve graphlet information as output.

Counting of subgraphs (e.g. motifs or graphlets) in a given network is the primary task of subgraph mining techniques [21][22][23]. Examples of network subgraph mining tools and techniques are mFinder [17][18], FANMOD [25], Kavosh [26], NetMode [27], ORCA [22], acc-Motif [28], and Co-evolving relational motifs [29]. In subgraph mining, counting a class of subgraph requires different approaches of graph theory. Usually a subgraph mining tools and techniques does (following) three steps. Firstly, it finds isomorphic subgraphs in a network.

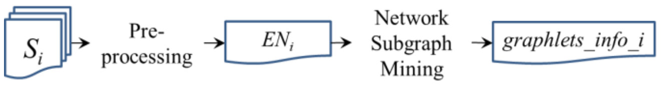


Fig 1. Subgraph mining of an evolving network EN_i for a state S_i.

Secondly, it groups them into classes. Thirdly, determine frequencies of the subgraph classes. We are interested to enhance such existing subgraph mining technique that can study system network evolution over time.

Subgraph mining for a network EN_i of a state S_i can retrieve graphlet information. Sequentially, we can retrieve graphlets information for all states one by one (locally inside each state). Our aim is to process a state series to identify the network evolution graphlets and complexities of an evolving system.

In next four sections, we elaborated the use of a novel technique of mining evolving networks, which discovers the hidden information for an evolving system. In the next section, we introduce definitions and equations that are useful to describe the proposed algorithm and its application on real-world evolving systems.

III. Proposed Definitions

This section introduces five fundamental definitions. To start with, we express a <u>*state series*</u> SS as a set $\{S_1, S_2 \ldots S_N\}$ at various time points such that S_i belongs to time t_i. Suppose we already used a subgraph mining technique on each state of a state series to retrieve graphlet information for each state. Based on this assumption, we describe the following definitions.

Definition 1: A <u>*network evolution subgraph*</u> is a subgraph that is common in the state series SS, such that it appears in multiple states of the system with changing frequencies. Depending upon the properties, the *network evolution subgraph* can be of three types:

- <u>*network evolution frequent subgraph*</u>: if subgraph is frequently occurring (means more than minimum support and minimum confidence) in a network;
- <u>*network evolution motif*</u>: if subgraph is statistically recurrent in a network; and
- <u>*network evolution graphlet*</u>: if subgraph is induced subgraph in a network.

For a *network evolution subgraph*, its frequencies keep on changing with states. Aggregate these frequencies to generate accumulated knowledge for a *network evolution subgraph* of a state series. On one hand, *frequent subgraphs* and *network motifs* are frequently (or statistically) occurring (or recurrent) as compared to a given threshold. On the other hand, graphlets are induced subgraphs, which provide frequencies that can be combined to form and represent its network. Thus, we are interested to accumulate knowledge of graphlets over a state series. This knowledge leads to define *aggregate frequency* for each retrieved *network evolution graphlets*.

Definition 2: Let G_j is a *network evolution graphlet* (NEG). The $Aggregate\_freq_j$ denotes <u>*aggregate frequency*</u> of a NEG G_j. Aggregation could be of many types, like mean, median, mode, sum, count, max, and min. The arithmetic mean of frequency over N states for a NEG G_j is given by

$$Aggregate\_freq_j = \frac{\sum_{i=1}^{N} freq_{ji}}{N} \qquad \text{-- (2)}$$

where, $freq_{ji}$ is the *frequency* of NEG G_j in state S_i such that *i* vary from integer 1 to N and *j* is constant. This means *i* is a variable for different states and *j* is constant for G_j.

We are interested to retrieve *network evolution graphlets* with its frequency in given set of evolving networks to make an aggregated information.

Definition 3: Network Evolution Graphlets information is a doubleton set of retrieved NEGs (as a subgraphs) and their *aggregate frequencies* $Aggregate\_freq_j$. The *NEGs information* is denoted as

$$NEGs\_info = < G_j, Aggregate\_freq_j > | \ 0 \leq m' \leq M \quad \text{-- (3)}$$

where, G_j is j^{th} NEG with aggregate frequency as $Aggregate\_freq_j$ and j is the enumeration of the NEG; such that m' is the number of retrieved NEGs (distinctly non-redundant graphlets) over all the states in SS.

We represent an evolving system as a set of evolving networks to retrieve graphlets. Collectively these graphlets are helpful to calculate the complexity of the evolving networks. We enhance the existing concept of cyclomatic complexity to calculate the system network complexity over time. To do this, we calculate and store the cyclomatic complexity (number of independent paths) of all possible graphlets. Usually, the cyclomatic complexity also considers the number of connected components, but in our case, cyclomatic complexity is calculated only for small-connected subgraphs (e.g. graphlets or motifs). The frequencies of graphlets are meaningful quantity that can be useful to calculate complexity for the network. Using cyclomatic complexity and subgraphs, we can quantify the complexities of an evolving system states. Every subgraph possesses a certain cyclomatic complexity. Thus, our technique is applicable to any system that can be represented as network.

Firstly, use the *graphlet information* of a state to calculate its System State Complexity.

Definition 4 System State Complexity (SSC): The SSC_i is the complexity of a state S_i in an evolving system. Numerically, it is the weighted arithmetic mean of the cyclomatic complexity over the frequencies of all retrieved graphlets, such that a frequency represents the weight. Its formula is given by the following equation

$$\text{System State Complexity of } S_i \ (SSC_i) = \frac{\sum_{j=0}^{m}(freq_{ji} \times C_j)}{\sum_{j=0}^{m} freq_{ji}} \quad \text{-- (4)}$$

where C_j denotes cyclomatic complexity and $freq_{ji}$ represents frequency for graphlet G_j of state S_i and m is count of retrieved graphlets. The *SSCs_info* denotes the SSCs information as

$$SSCs\_info = < S_i, SSC_i > | \ 0 \leq i \leq N \quad \text{-- (5)}$$

Secondly, use the *NEGs information* to calculate the Evolving System Complexity in the following definition.

Definition 5: Evolving System Complexity (ESC) is defined as the aggregated complexity of a state series SS for an evolving system. Numerically, it is the weighted arithmetic mean of the cyclomatic complexity over the aggregate frequencies for all NEGs, such that an aggregate frequency represents the weight. Its formula is given by the following equation

$$\text{Evolving System Complexity} = \frac{\sum_{j=0}^{m'}(Aggregate\_freq_j \times C_j)}{\sum_{j=0}^{m'} Aggregate\_freq_j} \quad \text{-- (6)}$$

where, $Aggregate\_freq_j$ denotes the *aggregate frequency* of a graphlet G_j over time, C_j denotes the *cyclomatic complexity* of the graphlet G_j and m' is count of retrieved NEGs.

The equation (2), (5), and (6) deduces the *average of non-zero SSCs* is equal to the ESC. Note the ESC is not equal to the average of SSCs. The SSC_i calculates complexity of an individual state, whereas ESC calculates complexity of a state series. We need both quantities because they express their own physical significance independent of each other.

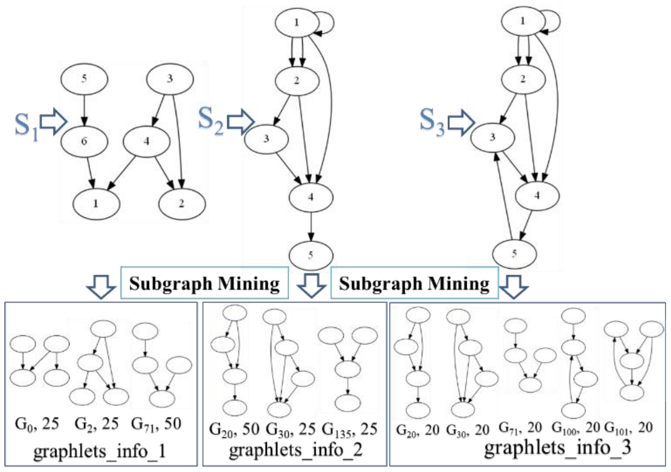

S_1
S_2
S_3
Subgraph Mining
Subgraph Mining
G_0, 25 G_2, 25 G_{71}, 50
graphlets_info_1
G_{20}, 50 G_{30}, 25 G_{135}, 25
graphlets_info_2
G_{20}, 20 G_{30}, 20 G_{71}, 20 G_{100}, 20 G_{101}, 20
graphlets_info_3

Figure 2. Illustrative example of network subgraph mining for a state series SS = {S_1, S_2, S_3} represented as three evolving networks.

An evolving system with less ESC has fewer interactions between its entities whereas an evolving system with high ESC has many complex interactions between its entities. If an evolving system has many subgraphs that have less complexity, then probably it has less ESC. If an evolving system has many subgraphs that have high complexity, then probably it has high ESC. The SSC can help to measure the change in a state. Using the ESC, we can compare the complexity of two evolving system of the same domain.

We show an illustrative example (in Figure 2) for subgraph mining of three network states. In Figure 2, suppose a state S_1 is pre-processed to make an evolving network $EN_1 = (E_1, C_1)$, where $E_1 = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ denotes a set of entities and $C_1 = \{(e_4, e_1), (e_4, e_2), (e_3, e_2), (e_5, e_6), (e_3, e_4), (e_6, e_1)\}$ denotes a set of directed evolving connection. Here, (e_4, e_1) represent a directed edge from entity e_4 to e_1. Similarly, S_2 and S_3 are also pre-processed to make EN_2 and EN_3 respectively. Thereafter, we do subgraph mining on all three evolving networks locally.

In Figure 2, for state S_1 graphlets (G_0, G_2, and G_{71}) has cyclomatic complexity ($C_0 = 1$, $C_2 = 2$, $C_{71} = 1$) and frequency ($freq_{0,1} = 25$, $freq_{2,1} = 25$, $freq_{71,1} = 50$). The resulting time-varying series of *system state complexity* for S_1 is calculated as $(25 \times 1 + 25 \times 2 + 50 \times 1) \div (25 + 25 + 50) = 1.25$ also inferable in Figure 3. The resulting *evolving system complexity* for state series SS = {S_1, S_2, S_3} is 1.816 also shown in Figure 3.

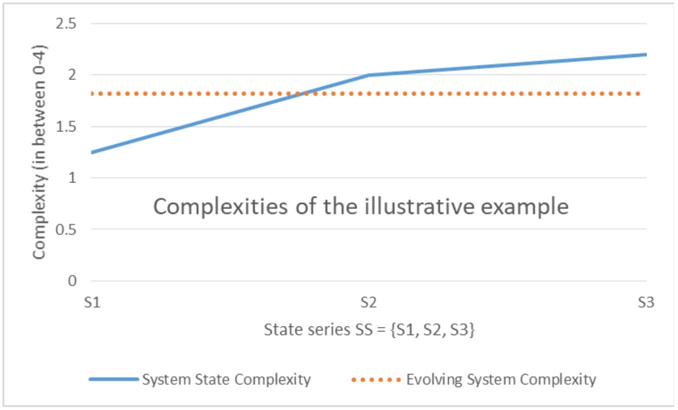


Figure 3. Illustrative example of complexity calculation for three states of an evolving system in Figure 2.

Next section presents the algorithm to retrieve *NEGs information*. It also presents detail about computational algorithms to calculate SSCs and ESC.

IV. PROPOSED: SYSTEM NETWORK COMPLEXITY

In this section, we present *System Network Complexity* (SNC) that analyses pre-evolved state series $\mathbb{SS}$ of an evolving system. The algorithms are useful to retrieve the evolving graphlets and complexities between inter-connected entities of an evolving system. Functionally, the discovered *Network Evolution Graphlets* (NEGs) is the subgraph of inter-connected entities occurring in the evolving networks over time. Functionally, a *System State Complexity* (SSC) represents complexity of a single state. Functionally, an *Evolving System Complexity* (ESC) represents aggregate complexity for a state series $\mathbb{SS}$ of an evolving system.

The NEGs, SSCs, and ESC can be retrieved with our proposed SNC algorithm. Search space of SNC is an evolving system that has N evolving states as a state series, $\mathbb{SS} = \{S_1, S_2 \ldots S_N\}$, stored in a repository. All the steps in the SNC are according to the guidelines of knowledge discovery and data mining - as shown in the Figure 4. The Figure 5 shows the flow chart of the *Algorithm SNC* that integrates the subgraph mining and evolution mining to discover the information about NEGs. The SNC outputs evolution information about the network evolution subgraphs in the form of *NEGs information* of an evolving system. Further, we used the NEG information to calculate SSCs and ESC.

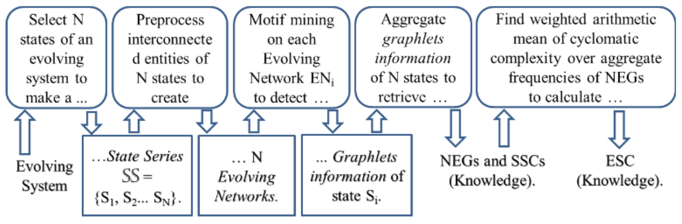

Select N states of an evolving system to make a ...
Preprocess interconnected entities of N states to create
Motif mining on each Evolving Network EN$_i$ to detect ...
Aggregate graphlets information of N states to retrieve ...
Find weighted arithmetic mean of cyclomatic complexity over aggregate frequencies of NEGs to calculate ...
Evolving System
...State Series $\mathbb{SS}$ = {S$_1$, S$_2$... S$_N$}.
... N Evolving Networks.
... Graphlets information of state S$_i$.
NEGs and SSCs (Knowledge).
ESC (Knowledge).

Figure 4. An overview of the process-artifact interaction in the System Network Complexity. Where, each rounded-rectangle shows a process and rectangle shows artifact created after the process. Each sentence starts from top and finishes at the artifact box. Where, the upward arrow denotes artifact goes to a process and downward arrow denotes artifact is generated from a process.

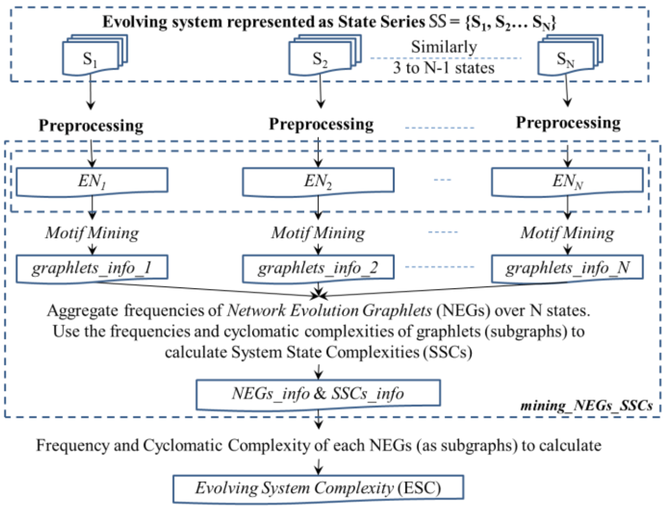

Evolving system represented as State Series $\mathbb{SS}$ = {S$_1$, S$_2$... S$_N$}
S$_1$
S$_2$
Similarly 3 to N-1 states
S$_N$
Preprocessing
Preprocessing
Preprocessing
EN$_1$
EN$_2$
EN$_N$
Motif Mining
Motif Mining
Motif Mining
graphlets_info_1
graphlets_info_2
graphlets_info_N
Aggregate frequencies of Network Evolution Graphlets (NEGs) over N states.
Use the frequencies and cyclomatic complexities of graphlets (subgraphs) to calculate System State Complexities (SSCs)
NEGs_info & SSCs_info
mining_NEGs_SSCs
Frequency and Cyclomatic Complexity of each NEGs (as subgraphs) to calculate
Evolving System Complexity (ESC)

Figure 5. An overview of flow-chart for the System Network Complexity.

Algorithm ***SNC****(repository)*

```
Initialize $i \in$ integer 1 to N
Retrieve N states of a state series $\mathbb{SS} = \{S_1, S_2 \ldots S_N\}$ stored in repository
1. evolvingNetworks = Preprocess(repository)
2. NEGs_info & SSCs_info
        = mining_NEGs_SSCs(evolvingNetworks)
3. ESC = Calculate_ESC(NEGs_info)
```

Next, we will present the three steps of SNC in detail to ensure their reproducibility. *Algorithm 1* ***Preprocess*** uses the N states of an evolving system stored in a repository such that each state S_i is pre-processed to make an EN_i, where i represent a state number varying from 1 to N. The pre-processing creates N evolving networks for N states.

Algorithm 1 ***Preprocess****(repository)*

```
For each state $S_i$ where $i \in$ integer 1 to N
    Detect an evolving network of inter-connected entities in $S_i$
    //Evolving network contains <sourceEntityIDs, targetEntityIDs>
    Denote the evolving network as $EN_i$
End For
Return evolvingNetworks
```

The *Algorithm 2* ***mining_NEGs_SSCs*** takes N *evolving networks* (as input) corresponding to the N states of an evolving system. The algorithm processes N *evolving networks* one by one i.e. each EN_i is processed locally, where 'i' varies from 1 to N. Thereafter, the algorithm retrieves *NEGs information* (as output) in two sub-steps.

- First, the network-subgraph mining algorithm processes each EN_i to detect graphlets information (*graphlets_info_i*) according to equation (1). To do so, we used HashMap that contains pair of (key, value) such that a set contains unique key and its value. A subgraph mining technique can be used as ***networkSubgraphMining***, which we described in Section II.

- Second, the algorithm aggregates the *graphlets_info* of a state series by calculating their *aggregate frequencies*. Let there are m' retrieved graphlets (unique and non-redundant) over all the states. This results in the *NEGs_info* as presented in equation (3).

Thereafter, the algorithm also calculates N *system state complexities* for all N states. The frequencies of retrieved graphlets (in *graphlets_info_i*) with cyclomatic complexities of all M graphlets are used to calculate *system state complexity* for each state. To do this, an array with pre-calculated cyclomatic complexity (C_j) for NEG (G_j) is used according to the equation (4). The SSC of all the N states are stored in a hash-map SSCs_info according to the equation (5).

Algorithm 2 ***mining_NEGs_SSCs****(evolvingNetworks)*

```
Initialize $j \in$ integer enumeration for graphlet $G_j$ such that $0 \le j \le M$
Initialize $i \in$ integer varying from 1 to N states
Initialize HashMap graphlets_info< $G_j$, $freq_{ji}$ >
Initialize HashMap NEGs_info< $G_j$, $Aggregate\_freq_j$ >
Initialize HashMap SSCs_info< $S_i$, $SSC_i$ >
```

Where, G_j is j^{th} *graphlet*, $freq_{ji}$ is *frequency* of G_j of state S_i, and $Aggregate\_freq_j$ is aggregate frequency of NEG G_j over a state series.

```
For each state $EN_i$ in evolvingNetworks where i varies from 1 to N
   graphlets_info_i < $G_j$, $freq_{ji}$ >= networkSubgraphMining($EN_i$)
   such that m is number of retrieved graphlets in graphlets_info_i
End For
For each m' graphlets $G_j$ of all states, where j varies as enumeration
and m' is the count of retrieved graphlets
Initialize float frequencySum = 0
Initialize integer $Aggregate\_freq_j$ = 0
  For each graphlets_info_i of $S_i$ where i varies from 1 to N
   frequencySum = frequencySum + $freq_{ji}$
  End For
  $Aggregate\_freq_j$ = frequencySum ÷ N
  Add tuple < $G_j$, $Aggregate\_freq_j$ > to NEGs_info
  such that m' is number of retrieved NEGs over state series SS
End For
Initialize array C[M] of cyclomatic complexity for all graphlets
For each graphlets_info_i of $S_i$ where i varies from 1 to N
Initialize float frequencySum = 0
Initialize float sumOfProducts = 0
  For each m $G_j$ in graphlets_info_i, where j varies as enumeration
   frequencySum = frequencySum + $freq_{ji}$
   sumOfProducts = sumOfProducts + { $freq_{ji} \times C_j$ }
  // where Cyclomatic complexity $C_j$ for graphlet $G_j$ at C[j]
  End For
  Float $SSC_i$ = sumOfProducts ÷ frequencySum
  Add tuple < $S_i$, $SSC_i$ > to SSCs_info
End For
Return NEGs_info & SSCs_info
```

The *Algorithm 3* ***Calculate_ESC*** uses the aggregate frequency ($Aggregate\_freq_j$) and cyclomatic complexity (C_j) of a j^{th} NEG (G_j) to calculate the ESC for all m' retrieved NEGs. We assign pre-calculated cyclomatic complexity (C_j) for all graphlets (G_j) in an array. Firstly, for each graphlet calculate "the *sum of products* between all cyclomatic complexity (C_j) and the aggregate frequency (*$Aggregate\_freq_j$*) of the NEG (G_j)". Second, for all the NEGs, also calculate "the *sum of frequencies* of NEGs". Divide the *sum of products* by the *sum of frequencies* to calculate the ESC, according to the equation (6).

Algorithm 3 ***Calculate_ESC(****NEGs_info***)**

```
Let m' is the number of retrieved graphlets in NEGs_info
Initialize $j \in$ integer enumeration for graphlets such that $0 \leq j \leq M$
Initialize float sumOfProducts = 0
Initialize array C[M] of cyclomatic complexity for NEGs ($G_j$)
$G_j \in j^{th}$ NEG in NEGs_info
Initialize $Aggregate\_freq_j$ = 0
$Aggregate\_freq_j \in$ aggregate frequency of $G_j$ in NEGs_info
For each m' NEGs $G_j$ over all states, where j varies as enumeration
   sumOfProducts = sumOfProducts + { $Aggregate\_freq_j \times C_j$ }
   //where Cyclomatic complexity $C_j$ for graphlet $G_j$ at C[j]
   frequencySum = frequencySum + $Aggregate\_freq_j$
End For
Float ESC = sumOfProducts ÷ frequencySum
Return ESC
```

The computational complexity of the SNC algorithm can be given in the following way. Firstly, the computational complexity of *Algorithm 1 Preprocess* is dependent upon the system domain and technique for pre-processing system state to make evolving networks. Secondly, the computational complexity of the *Algorithm 2 mining_NEGs_SSCs* is $O(N\times\lambda) + 2O(N\times m')$, where N is number of states, m' is the number of retrieved NEGs over all states, and λ is the complexity of subgraph mining algorithm. Thirdly, the computational complexity of *Algorithm 3 Calculate_ESC* is O(m'). Thus, the SNC complexity mainly depends upon *Algorithm 2 mining_NEGs_SSCs*, which majorly depends on $O(N\times\lambda)$ i.e. number of states (N) times complexity of subgraph mining (λ).

Based on the algorithms presented in this section, we describe our prototype tool in the next section.

V. SNC-Tool

Based on the SNC algorithm, we developed a Java-based tool named as *SNC-Tool*, which is executable on machine enabled with JRE and JDK 7th version or higher. There are many existing subgraph mining algorithms and open source tools. In *SNC-Tool*, we used the *acc-Motif* tool [51] as network subgraph mining algorithm developed by Meira et al. [28], which detects accelerated motif (acc-Motif) using combinatorial techniques. We opted for the acc-Motif because it is an efficient open-source tool, which is implemented in Java (the language opted for our SNC-Tool). Additionally, the acc-Motif builds upon old tools and techniques. The acc-Motif (that we used for network subgraph mining) has following three computational time complexities $\lambda = O(a(G)e)$, $O(e^2)$, and $O(ne)$ for the detection of subgraph size 3, 4 and 5 in directed graphs, where a(G) is the arboricity of graph G(V, E), n is |V| vertices, and e is |E| edges.

The *SNC-Tool* has three components '*PreProcessing'*, '*Mining_NEGs_SSCs',* and '*EvolvingSystemComplexity*'. First component is based on *Algorithm 1 Preprocess*(*repository*). Second component is based on the proposed *Algorithm 2 mining_NEGs_SSCs*(*evolvingNetworks*). Third component is based on *Algorithm 3 Calculate_ESC*(*NEGs_info*). The next section, describes the use of SNC-Tool by doing experiments on six different evolving systems.

VI. Experiments on Evolving Systems

This section describes an empirical evaluation of SNC-Tool for evolving systems available on open internet repositories of four domains: software, natural language, retail market, and IMDb. We also discuss a detailed application of SNC for each domain and demonstrate experiments (using SNC-Tool) on six evolving systems of the four domains as given in the Table I. Additionally, Table I describes the source and target entities for the type of connections to create a type of network. The type of connection and the type of network depend upon the domain of the evolving system. Some type of network has specific name in their domain like *call graph* in software and *word network* in natural language. Some type of networks do not have specific name, thus, we name them by our-self like: *purchase network*, *positive sentiment network*, and *negative sentiment network*.

Description of the Table II is as follows. First column contains name of the system used in the experiment. Second column contains number of states used in the experimentation. Third column contains the number of entities. Fourth column contains *average number of neighbours* (entities), it is a commonly used property of network or graph. Fifth column contains number of NEGs. Sixth column contains calculated value of Evolving System Complexity (ESC).

TABLE I. DOMAIN DESCRIPTION OF THE SIX EVOLVING SYSTEMS.

Domains of Evolving System	**Evolving Systems**	**“Source” and “Target” Entities**	**Type of Connections**	**Type of network**
(A) Evolving Software System	Hadoop HDFS-Core[1]	“Caller” and “Callee” Procedures	Procedural calls	Call graph
(B) Evolving Natural language systems	Bible Translation[2]	Words in “Source biblical language” and “English variant” languages	Translations	Words Network
	Multi-sport Events[3]	Words in “Titles” (name) and “Scopes” (region) of events	Regional names	Words network
(C) Evolving Retail Market System	Frequent Market Basket[4]	Words in “Product description” and Words in “Product description”	Purchases	Purchase network
(D) Evolving IMDb movie genre systems[5]	Positive sentiment[6] of movie genres[5]	“Positive words in names” and “genres” of movies	Sentiments	Positive sentiment network
	Negative sentiment[6] of movie genres[5]	“Negative words in names” and “genres” of movies	Sentiments	Negative sentiment network

1. https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-hdfs 2016.
2. https://en.wikipedia.org/wiki/List_of_English_Bible_translations Oct 2016.
3. https://en.wikipedia.org/wiki/List_of_multi-sport_events Oct 2016.
4. https://archive.ics.uci.edu/ml/datasets/Online+Retail Oct 2016.
5. http://www.imdb.com/interfaces/ Oct 2016.
6. https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html June 2017

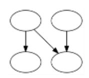
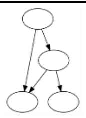
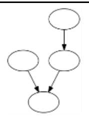
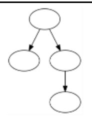
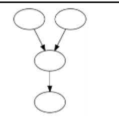
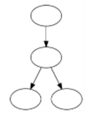
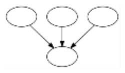
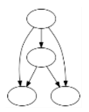
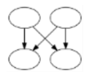
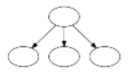
Evaluation of NEGs: We retrieved NEGs information for each evolving system using the proposed mining algorithm that uses the set of evolving networks along with input parameters. We used parameters to fine-tune the retrieval of subgraph according to the desired accuracy. Although in each experiment we retrieved many NEGs, but we present significant NEMs (network evolution motifs with high frequencies) in the Table III, which also has complexities of those subgraphs (size 4).

TABLE II. EXPERIMENTAL RESULTS ON THE EVOLVING SYSTEMS.

Evolving Systems	**N**	**# entities**	**Average # neighbour**	**# NEGs**	**ESC***
HDFS-Core[1]	15	3129	2.166	24	1.0054
Bible Translation[2]	13	246	1.456	17	1.456
Multi-sport Events[3]	13	141	1.786	10	1.00487
Frequent Market Basket[4]	13	118	8.002	34	1.33888
Positive sentiment[6] of movie genres[5]	16	284	2.661	4	1.03050
Negative sentiment[6] of movie genres[5]	16	510	3.303	20	1.03683

# stands for “number of”. *Range of ESC is in between 1 to 4 because cyclomatic complexity of subgraph-size (subgraph-4) is in between 1 to 4.

TABLE III. MOST SIGNIFICANT RETRIEVED NEMS (SUBGRAPHS).

$M_0, C_0 = 1$	$M_2, C_2 = 2$	$M_{71}, C_{71} = 1$	$M_{87}, C_{87} = 1$	$M_{135}, C_{135} = 1$
				
$M_{156}, C_{156} = 1$	$M_{178}, C_{178} = 1$	$M_{185}, C_{185} = 3$	$M_{192}, C_{192} = 2$	$M_{198}, C_{198} = 1$
				

Evaluation of SSCs and ESC: We used the retrieved information about subgraphs of an evolving system to calculate its complexity. Using *frequencies* and *cyclomatic complexities* of the graphlets (in *graphlets_info*), we computed SSC for each state according to the equation (4). Using *aggregate frequencies* and *cyclomatic complexities* of the *NEGs,* we computed ESC for a state series according to the equation (6). We calculated ESC for all evolving systems, which are given in Table II. Since every *NEG* (subgraph) has exactly 4 nodes and all such subgraphs have complexity in between 1 to 4. Therefore, SSCs and ESC values will range from 1 to 4. Low complexities of resulting NEGs produce low SSCs and ESC.

The six experimental results are shown as six time series plots. In each plot, the x-axis represents state series $\mathbb{SS}$ of N states where each instance is a state (S_i) and y-axis represents the complexity of each state. In the time series plots (of Figure 6, 7, 8, 9, 10, and 11), there are two time series for various states. First, the time-varying series shows the *system state complexity* (SSC) of each states. Second, the constant time series shows the ESC of the whole system state series. For example, to read the time series in Figure 6, the HDFS-core version 2.2.0 (a state) has SSC = 1.00625 and for a state series it has ESC = 1.0054.

Now we describe the experiments on six evolving systems of four domains.

(A) Evolving software systems [30] are stored and maintained as a state series $\mathbb{SS}$ at some software repository. A state series of software can be referred as a version series because a software state is referred as a software version.

We collected 15 Hadoop-HDFS-Core[1] versions (states) of jars. Many open-source tools are available to construct *call graph* of a software. Software developed in different programming languages has different tools to do this. The call graph contains procedure call information stored in the form of a network. It is a directed graph to describe procedure-call (or dependency) relationships. Each line of a *call graph* file contains two nodes (as procedures) to represent a directed edge (as call) from the first node to the second node. Where, the first node represents caller procedure and the second node represents callee procedure. A HDFS version series of 15 versions processed to a series of 15 *call graphs* (networks). During pre-processing, a same *procedureID* keeps track for a procedure in different modules of repository. We observed the call graphs of procedure calls by identifying the graphlets that gives us an insight of induced subgraphs. The induced subgraph helps in the calculation of complexity of the evolving software. The time series plot in Figure 6 shows the time-varying complexity of each version (state) and ESC (1.0054).

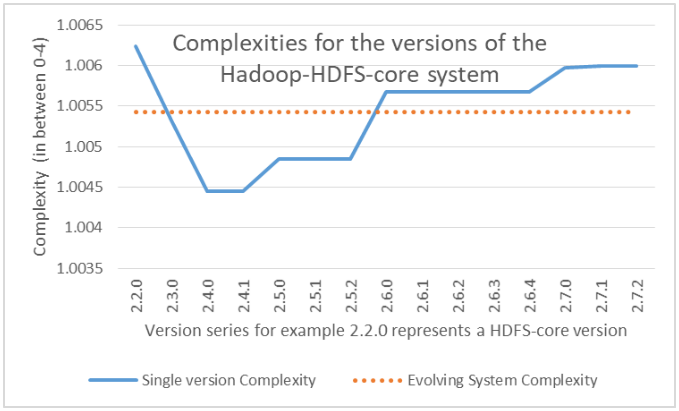


Figure 6. The time series representation of SSCs and ESC for Hadoop HDFS-core over 15 versions as a version series.

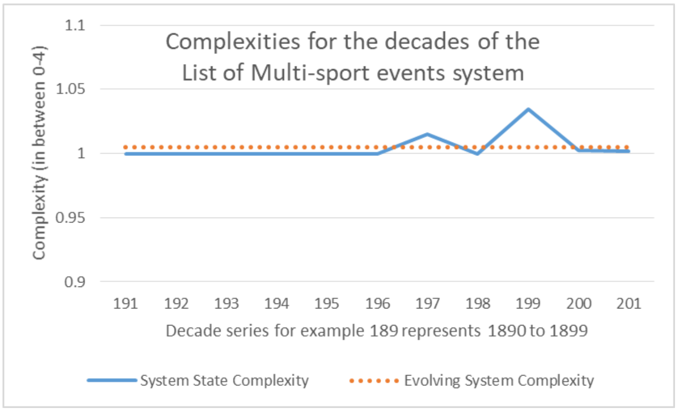


Figure 7. The time series representation of SSCs and ESC for multi-sport events over 11 decades as a decade series. Note, out of 13 decades we omitted two decades (180's and 190's) from the x-axis because their resulting complexity is undefined due to insufficient data.

We got "M_{178}" as the most significant *NEM*, where M_{178} suggests a subgraph where three procedures are calling a procedure, which depicts less complex subgraph. This high frequency of low complex subgraph leads to the low ESC of the Hadoop HDFS-core.

(B) Evolving natural language systems: We had chosen two evolving systems that are available on Wikipedia: list of bible translations[2] and list of multi-sport events[3]**.**

(i) List of Bible translations[2] contain table that has "source biblical languages" (like Hebrew, Aramaic, and Greek) to the various "English variant languages" (like Modern English and Old English). To generate the evolving state series SS we combined the tables of "incomplete Bibles", "partial Bibles" and "complete Bibles" given in the link[2]. We made evolving networks from connections between entities (words) in columns of 'Source biblical language' and 'English variant'**,** such that each network is for a century. There are 13 evolving networks for data of 13 centuries. The time series plot in Figure 7 shows the time-varying complexity of each state (century) and ESC (1.45).

We can observe three points from the Figure 7. First, the figure shows spike in between 8th to 10th centuries because there are many translations happened in this period. Second, the figure shows increase in between 11th to 14th century this is due to increase in number of translation in this period. Third, the figure shows, after 14th century the bible translation complexity is almost constant due to stable number of translations in this period.

(ii) List of multi-sport events[3]: We made evolving networks from connections between entities (word) in two columns: 'Title' (name) and 'Scope' (regional, international, and provinces) of an event, such that a network is for a decade (10 years). There are total 13 evolving networks are retrieved for 13 decades. The time series plot in Figure 8 shows complexity for connections between entities 'Title' and 'Scope' of multi-sport events. In the figure, first time series is for the time-varying complexity of each state (decade) and second time series is for ESC (1.004). We can observe from the Figure 8 that the figure shows almost constant complexity between 191 to 196's decade and 200 to 201's decade because there are few target nodes of scopes.

(C) Evolving retail market system[4] is a dataset of retail market [31] for UK based registered non-store online retail transaction between 01/12/2010 and 09/12/2011. To make evolving network we followed following steps. First, we make a collection of market basket from products bought by customers with their IDs during each month. Second, to make frequent market basket of products, we kept a product as node in the network only if it is sold atleast 10 months. We used these frequent products to generate the evolving networks for products bought by customers, such that connections between words in column 'Description' of products. We generated 13 such evolving networks for 13 months such that each network is for a month. The time series plot in Figure 9 shows the time-varying complexity of each state (month) and ESC (1.33).

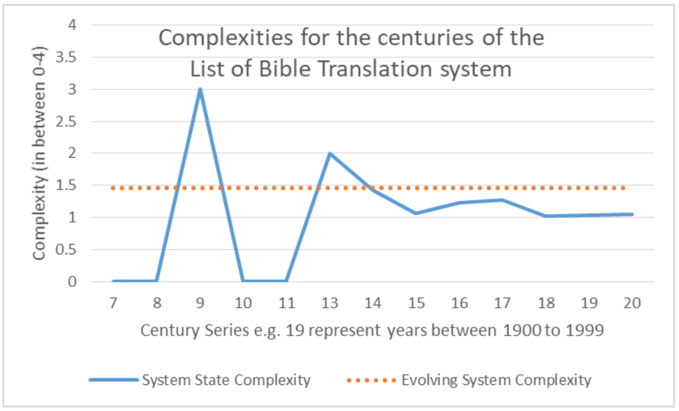


Figure 8. The time series representation of SSCs and ESC for list of bible translation over 13 centuries as a century series.

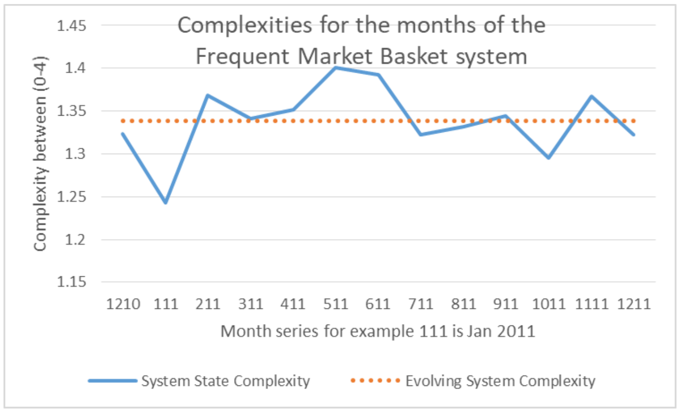


Figure 9. The time series representation of SSCs and ESC for frequent market basket over 13 months as a month series.

We can observe two points from the Figure 9. First, the figure shows high complexity in between 5th to 6th month of 2011 (i.e. 511 to 611) because of many transactions happened in this period. This is may be due to “summer season in Europe promote more shopping”. Second, the figure shows two spikes, firstly sudden decrease in month January 2011 (111) this is due to “sale decreases in December just after the Christmas”, and secondly sudden increase in November 2011 (1111) month this is contrarily due to “sale increases in November just before the Christmas”.

(D) Evolving IMDb movie genre system[5]: We collected the movie and genre data of IMDb for 16 decades since 1870’s until Oct 2016. In Figure 10 and 11, out of 16 decades we omitted three decades (187’s, 188’s, and 202’s) from the x-axis because their resulting complexity is undefined due to insufficient data. We used list[6] of positive and negative words, which was created and used by Minqing and Bing et al. [32][33].

(i) Evolving system 5: We used positive sentiment words in movie names and genres of IMDb in this experiment. While generating the evolving networks, we used specifically used only positive words in movie names. The Figure 10 shows experimental results for entities (words) connections between 'positive words in movie names' and 'genres' in the IMDb dataset. The time series plot in Figure 10 shows the time-varying complexity of each state (month) and ESC (1.03).

(ii) Evolving system 6: We used negative sentiment words in movie names and genres of IMDb in this experiment. While generating the evolving networks, we specifically used only negative words in movie names. The Figure 11 shows experimental results for connections between entities 'negative words in movie names' and 'genres' in the IMDb dataset. The time series plot in Figure 11 shows the time-varying complexity of each state (month) and ESC (1.03).

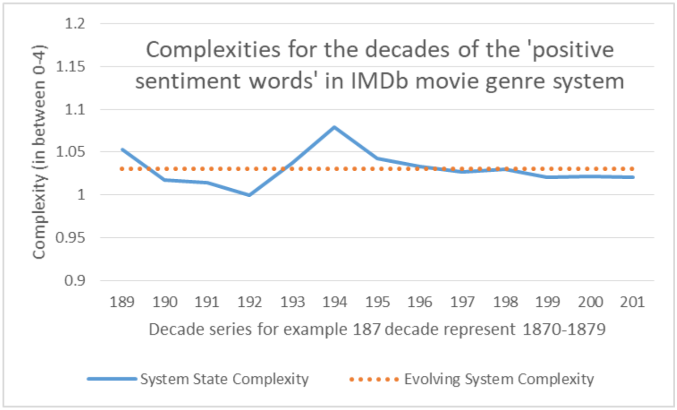


Figure 10. The time series representation of SSCs and ESC for positive sentiment IMDb system over 13 decades as a decade series.

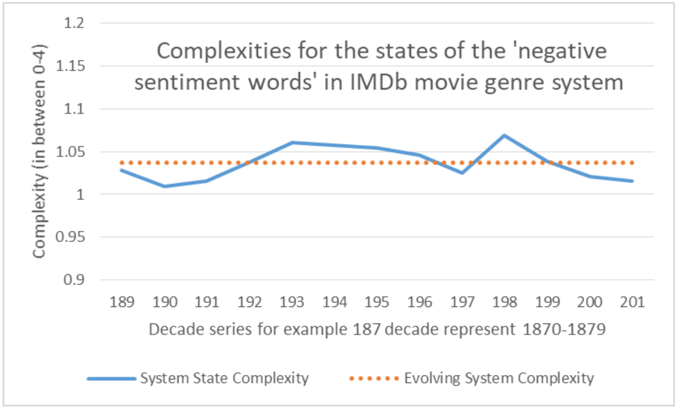


Figure 11. The time series representation of SSCs and ESC for negative sentiment IMDb system over 13 decades as a decade series.

We made two observations from the Figure 10. First, the SSCs are virtually uniform in between 198’s to 201’s decades because there is large number of similar genre movies in this period. Second, SSCs varies for decades between 189’s to 195’s because there are fewer interactions in less number of different genres. In Figure 11, SSCs varies for all decades because of fewer interactions due to less number of different genre movies data. From Figure 10 and 11, we infer values of both ESC and SSCs are almost equal to 1.04 for both the systems.

Observations:

- Using time series plots, we can measure the change in the complexities between two states in an evolving system.
- We got M_0, M_{178} and M_{198} as the most common and significant NEMs in the most of the evolving systems. The M_{178} suggests a subgraph where a single entity connects to three different entities, which depicts less complex subgraph.
- When two evolving systems have different pre-processing to generate their evolving networks, then they are not comparable to each other based on their SSCs and ESC.
- When different evolving systems have same pre-processing semantics to generate their evolving networks, then they are comparable to each other based on their SSCs and ESC. For example, we can compare the positive and negative sentiment of the movies name with genres because of the same pre-processing semantics to generate evolving networks. The Table II quantitatively shows that “the movies promote negativity more than positivity” because use of negative words (i.e. entities) is more than the positive words (i.e. entities) in movie names. The Table II also depicts, the average number of neighbours, NEGs (subgraphs), and ESC of negativity is more than positivity in movie names.

A threat to validity**:** The SNC depends upon network subgraph mining, thus pros-cons of subgraph mining are pros-cons of SNC algorithm. As a limitation, our algorithm depends on the efficiency of subgraph mining technique. Although our SNC algorithm can work with any subgraph-mining technique, but the choice of the subgraph-mining technique influences the performance of the SNC. Thus, the choice of subgraph mining tool should be prudent, which depends on the required output.

Contributions from experiments:

1. We demonstrated the integration of graphlet and evolution information while applying SNC on six state series of six evolving systems, which are collected from open-internet repositories of four domains. The SNC-Tool generated SSCs and ESC information about each evolving system.
2. Using SSCs, we inferred complexity changes happened in an evolving system. Using SSCs and ESC, we compared two evolving systems (if they are of same domain with same pre-processing).

VII. Related Works & Discussions

This section presents and discusses existing state-of-the-art. We begin with application of motifs in evolving and temporal networks. Braha and Bar-Yam [34] studied denser motifs on e-mail dataset in which each snapshot network represents contacts aggregated over a day. Zhao et al. [35] studied the temporal annotations (e.g., timestamps and duration) of historical communications graphlet in social networks using communication motifs and their maximum flow. Jurgens and Tsai [36] proposed a method to represent Wikipedia revision history as a temporal bipartite graph of editor interactions, which identifies significant author interactions as network motifs of diverse editing behaviours. Yan and Guo [37] proposed evolving gene regulatory networks (GRNs) based self-organizing robotic to generate evolving network motifs for path detection in unknown environment autonomously. Bhattacharya et al. [38] applied network motifs of software system to capture, analyse, and infer software properties.

Hulovatyy et al. [39] presented a dynamic graphlets based analysis of temporal networks by capturing inter-snapshot relationships. Recently, Martin et al. [40] proposed two work: (a) REConstruction Rate (REC) to determine the rate of graphlet similarity between different states of a network and (b) REC Graphlet Degree (RGD) to identify the subset of nodes with the highest topological variation.

Kovanen et al. [41] proposed a technique to detect *temporal motif* based on significant, intrinsically dynamic, mesoscopic structures, and graphlets in temporal networks. Paranjape et al. [42] presented temporal network motifs as induced subgraphs on sequences of temporal edges; they also designed algorithms for counting temporal motifs. In contrast to these approaches, we presented network evolution subgraphs that are further used to calculate the complexity of evolving system states.

Pincus [47] described approximate entropy (ApEn) as mathematical family of formulas and statistics, which quantify the concept of changing complexity. Rosen [48] described the complexity as a property to interact with systems around us that are continually changing. Different techniques are applied to measure the complexity of software system [49] and ecological system [50]. Whereas, our subgraph mining approach aids to calculate complexity based on McCabe [24] technique.

Earlier Pržulj [43], Yaveroğlu et al. [44], Ali et al. [45], and Wegner et al. [46] presented approaches to analyze and compare networks. The four approaches deals with comparison of networks, whereas our approach does not intends to compare two graph directly. Alternatively, we aim to compare time-varying complexities (SSCs) and ESC of two evolving systems modelled as evolving network series. Possible applications of our work is to study complexity of domains like social media [34], communication motifs [35], natural language processing [36], robot path [37], and software systems [38].

VIII. Conclusions

In this paper, the *graphlets information* is used to calculate the *System State Complexity* (SSC) for each state S_i. The *network evolution graphlets* (NEGs) information is used to calculate the *Evolving System Complexity* (ESC) for a state series SS. Both SSC (of a state) and ESC (of a state series) are interesting and non-obvious information.

We implemented the SNC as a prototype SNC-Tool, which is used to conduct experiments on six open-internet based evolving systems. The tool retrieved *network evolution subgraphs* such as NEGs and *network evolution motifs* (NEMs). We also calculated the complexity of individual states (as SSCs) and complexity of a state series (as ESC). The SSCs are useful to compare between two states of an evolving system. The ESC is useful to compare between two different evolving systems of the same domain. This provides insightful and actionable information about an evolving system.

Our SNC algorithm computation is novel with respect to study of system complexities. In addition to the mining NEGs, we computed the SSCs and ESC over a state series. The merit of our work is that it is an extension of the popular and well-used cyclomatic complexity technique of McCabe [24]. Our technique to measure complexity is significantly different from existing techniques. Best to our knowledge, we are the first introduce SNC theory that includes SSC and ESC.

In future, this kind of system evolution analysis can be done using other evolution mining or learning techniques. We plan to use other techniques for studying system evolution. Further, we plan to execute our tool on other types of evolving systems.

Acknowledgments

We would like to thank graduate student Vraj Shah of IIT Indore for his help in the beginning of this work.

References

[1] Ross Adam M., Donna H. Rhodes, and Daniel E. Hastings. "Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value." *Systems Engineering* 11.3 (2008): 246-262.

[2] Frost Susan A., and Mark J. Balas. "Evolving Systems and Adaptive Key Component Control." (2009).

[3] Angelov Plamen, and Nik Kasabov. "Evolving intelligent systems, eIS." *IEEE SMC eNewsLetter* 15 (2006): 1-13.

[4] Palpanas Themis. "Data series management: The next challenge." *IEEE 32nd Int. Conf. on Data Engineering Workshops* (ICDEW). IEEE, 2016.

[5] Ditzler Gregory, et al. "Learning in nonstationary environments: a survey." *IEEE Computational Intelligence Magazine* 10.4 (2015): 12-25.

[6] Böttcher Mirko, et al. "On exploiting the power of time in data mining." *ACM SIGKDD Explorations Newsletter* 10.2 (2008): 3-11.

[7] Takaffoli Mansoureh, et al. "Community evolution mining in dynamic social networks." *Procedia-Social and Behavioral Sciences* 22 (2011): 49-58.

[8] Abraham Tamas, and John F. Roddick. "Incremental meta-mining from large temporal data sets." *Int. Conf. on Conceptual Modeling*. Springer Berlin Heidelberg, 1998.

[9] Zadeh Lotfi A. "Time-varying networks, I." *Proceedings of the IRE*49.10 (1961): 1488-1503.

[10] Aronson Jay E. "A survey of dynamic network flows." *Annals of Operations Research* 20.1 (1989): 1-66.

[11] Holme Petter, and Jari Saramäki. "Temporal networks." *Physics Reports* 519.3 (2012): 97-125.

[12] Kivelä Mikko, et al. "Multilayer networks." *J. of Complex Networks* 2.3 (2014): 203-271.

[13] Aggarwal Charu, and Karthik Subbian. "Evolutionary network analysis: A survey." *ACM Computing Surveys* 47.1 (2014): 10.

[14] Inokuchi Akihiro et al. “An apriori-based algorithm for mining frequent substructures from graph data.” *European Conf. on Principles of Data Mining and Knowledge Discovery*. Springer Berlin Heidelberg, 2000.

[15] Kuramochi Michihiro, and G. Karypis. “Frequent subgraph discovery.” *IEEE Int. Conf. on Data Mining* (ICDM). IEEE, 2001.

[16] Yan Xifeng and Jiawei Han. “gspan: Graph-based substructure pattern mining.” *IEEE Int. Conf. on Data Mining* (ICDM). IEEE, 2002.

[17] Milo R., Shen-Orr S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon U. “Network Motifs: Simple Building Blocks of Complex Networks.” *Science*. 298, 824—827 (2002).

[18] U. Alon, “Network motifs: theory and experimental approaches” *Nature Rev. Genetics* (2007) 450–461.

[19] Pržulj Natasa, Derek G. Corneil, and Igor Jurisica. "Modeling interactome: scale-free or geometric?." *Bioinformatics* 20.18 (2004): 3508-3515.

[20] Milenkoviæ Tijana, and Nataša Pržulj. "Uncovering biological network function via graphlet degree signatures." *Cancer informatics* 6 (2008): 257.

[21] Ahmed Nesreen K., et al. "Graphlet decomposition: Framework, algorithms, and applications." *Knowledge and Information Systems* 50.3 (2017): 689-722.

[22] Hočevar Tomaž, and Janez Demšar. "A combinatorial approach to graphlet counting." *Bioinformatics* 30.4 (2014): 559-565.

[23] Ahmed Nesreen K., Theodore L. Willke, and Ryan A. Rossi. "Estimation of local subgraph counts." *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016.

[24] McCabe Thomas J. “A complexity measure.” *IEEE Trans. on Soft. Eng*. 4 (1976): 308-320.

[25] Wernicke Sebastian, and Florian Rasche. “FANMOD: a tool for fast network motif detection.” *Bioinformatics* 22.9 (2006): 1152-1153.

[26] Kashani Zahra RM, et al. “Kavosh: a new algorithm for finding network motifs.” *BMC Bioinformatics* 10.1 (2009): 318.

[27] Li Xin, et al. “NetMODE: network motif detection without Nauty.” *PloS One* 7.12 (2012): e50093.

[28] Meira Luis AA, et al. “acc-Motif: accelerated network motif detection.” *IEEE/ACM Trans. on Computational Biology and Bioinformatics* (TCBB) 11.5 (2014): 853-862.

[29] Ahmed Rezwan, and George Karypis. “Algorithms for mining the coevolving relational motifs in dynamic networks.” *ACM Trans. on Knowledge Discovery from Data* (TKDD) 10.1 (2015): 4.

[30] Tom Mens, Alexander Serebrenik, and Anthony Cleve. “Evolving Software Systems.” *Springer Publishing Company, Incorporated*. 2014.

[31] Chen Daqing, Sai Laing Sain, and Kun Guo. "Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining." *J. of Database Marketing and Customer Strategy Management* 19.3 (2012): 197-208.

[32] Hu Minqing, and Bing Liu. "Mining and summarizing customer reviews." *10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data mining*. ACM, 2004.

[33] Liu Bing, Minqing Hu, and Junsheng Cheng. "Opinion observer: analyzing and comparing opinions on the web." *14th Int. Conf. on World Wide Web*. ACM, 2005.

[34] Braha Dan, and Yaneer Bar-Yam. "Time-dependent complex networks: Dynamic centrality, dynamic motifs, and cycles of social interactions." *Adaptive Networks*. Springer Berlin Heidelberg, 2009. 39-50.

[35] Zhao Qiankun, et al. "Communication motifs: a tool to characterize social communications." *19th ACM Int. Conf. on Information and knowledge Management*. ACM, 2010.

[36] Jurgens David, and Tsai-Ching Lu. "Temporal Motifs Reveal the Dynamics of Editor Interactions in Wikipedia." ICWSM. 2012.

[37] Meng Yan, and Hongliang Guo. "Evolving network motifs based morphogenetic approach for self-organizing robotic swarms." *Conference on Genetic and Evolutionary Computation*. ACM, 2012.

[38] Bhattacharya Pamela, et al. “Graph-based analysis and prediction for software evolution.” *34th Int. Conf. on Soft. Eng.*. IEEE Press, 2012.

[39] Hulovatyy Yuriy, Huili Chen, and T. Milenković. "Exploring the structure and function of temporal networks with dynamic graphlets." *Bioinformatics* 31.12 (2015): i171-i180.

[40] Martin Alberto JM, et al. "Graphlet Based Metrics for the comparison of gene regulatory networks." *PloS one* 11.10 (2016): e0163497.

[41] Kovanen Lauri, et al. "Temporal motifs in time-dependent networks." *J. of Statistical Mechanics: Theory and Experiment* 2011.11 (2011): P11005.

[42] Paranjape Ashwin, Austin R. Benson, and Jure Leskovec. "Motifs in temporal networks." *10th ACM Int. Conf. on Web Search and Data Mining*. ACM, 2017.

[43] Pržulj Nataša. "Biological network comparison using graphlet degree distribution." *Bioinformatics* 23.2 (2007): e177-e183.

[44] Yaveroğlu Ömer Nebil, et al. "Revealing the hidden language of complex networks." *Scientific reports* 4 (2014): 4547.

[45] Ali Waqar, et al. "Alignment-free protein interaction network comparison." *Bioinformatics* 30.17 (2014): i430-i437.

[46] Wegner Anatol E., et al. "Identifying networks with common organizational principles." *arXiv preprint arXiv:1704.00387*(2017).

[47] Pincus Steven M. "Approximate entropy as a measure of system complexity." *National Academy of Sciences* 88.6 (1991): 2297-2301.

[48] Rosen Robert. "Complexity and system descriptions." *Facets of Systems Science*. Springer US, 1991. 477-482.

[49] A. M. Abdullah, et al. "Modification of standard function point complexity weights system." *J. of Systems and Software* 74.2 (2005): 195-206.

[50] G. Jacques, et al. "Emergence and complex systems: The contribution of dynamic graph theory." *Ecological Complexity* 31 (2017): 34-49.

[51] acc-Motif: Accelerated Motif Detection, http://www.ft.unicamp.br/docentes/meira/accmotifs/ accessed Jun 2017.

Animesh Chaturvedi is working towards the PhD degree at the IIT Indore. He received the BEng degree from IET - DA University, and the MTech degree from Indian Institute of Information Technology, Design and Manufacturing, Jabalpur. To do research, he declined non-research based job offers of two MNC’s at early 20’s of his life. He did research work in Motorola, Arris, and IIT-Kanpur. He has been reviewer for IEEE TCC, IEEE TETC, IEEE TBD, and IEEE SJ. His research interest is in Machine Learning, Data mining, SOA, Cloud computing, Evolving systems, Software Maintenance and Evolution.

Aruna Tiwari is an Associate Professor of Computer Sci. and Eng. at IIT Indore, since 2012. She received BEng and MEng degrees in Computer Eng. Her PhD degree is in Computer Sci. & Eng. Her research interests include soft computing, neural network, fuzzy clustering, evolutionary computation, genome analysis, and health-care applications. Her research group is working on these techniques to accomplish goals of data mining, machine learning, big data analytics, and hardware realization. She has many publications in peer-reviewed journals (of IEEE Transactions, Elsevier, and Springer), international conferences, and book chapters. She has been reviewer for many reputed journals & conferences. She has research collaboration with CSIR CEERI Pilani and Indian Institute of Soyabean Research (Indian Council of Agriculture & Research).