minStab: Stable Network Evolution Rule Mining for System Changeability Analysis

Article *in* IEEE Transactions on Emerging Topics in Computational Intelligence · April 2019

DOI: 10.1109/TETCI.2019.2892734

Animesh Chaturvedi
Indian Institute of Information Technology, Design and Manufacturing Jabalpur

minStab: Stable Network Evolution Rule Mining for System Changeability Analysis

Animesh Chaturvedi, Aruna Tiwari and and Nicolas Spyratos

***Abstract*— Growing number of evolving systems creates demand for system evolution analysis with modern computational intelligence algorithms and tools. In this paper, we introduce new measures of *stability* and *changeability* for system evolution analysis over time. We proposed a *Stable Network Evolution Rule Mining* (SNERM) and a *Changeability Metric* (CM) for an evolving system. For this, we use two different characteristics of Network Evolution Rules (NERs). First, given a network of a system state S_i, we call a NER *interesting* in S_i if its support and confidence exceed given thresholds (minimum support and minimum confidence). Second, given a set of networks for a set of states SS, we define the *stability* of a NER to be the percentage of states in SS in which the rule is interesting. We call a NER *stable* in SS if its stability exceeds a given threshold named as *minimum stability* (minStab). Based on this, we developed an intelligent tool, which is used for experiments on evolving systems. We applied our approach to a number of real-world systems including: software system, natural language system, retail market system, and IMDb system. It results Stable NERs and Changeability Metric value for each evolving system.**

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

I. Introduction

COMPUTATIONAL INTELLIGENCE (CI) intends to create an approach, phenomenon, algorithm, or tool, which behaves intelligently. Usually, real world systems has complex environment, which make agents to change and evolve a system with time. Based on fundamentals of well-known association rule mining, we aim to create an intelligent algorithm and tool, which helps to study the system evolution and changeability.

An evolving system [1][2][3][4][5] is a complex system that evolves with time and have a number of different states. We consider such a system with two assumptions: (a) the system contains a number of distinct entities and (b) each entity might be connected to zero, one or more other entities over time. Thus, this creates an evolving network [6] (or dynamic network) of interconnected entities. We shall refer to this evolving network as the *system network* of entities.

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.

Nicolas Spyratos is with the University of Paris-Sud, Paris, France. E-mail: spyratos@lri.fr

A basic characteristic of an evolving system is that one or more connections present in a state S_a might not be present in another state S_b, while connections present in S_b might not present in state S_a. Every system has many entities that are transformable to make a dynamic database for the set of states. The various dynamic databases of system states can be stored and managed in a repository. The dynamic databases can be nonstationary data [7] in context of machine learning.

An example of evolving system is an *evolving software system* [8] under maintenance phase, which is a software system that evolves over time. It might be in different states as time passes, and its state is referred as software version. Indeed, a software system usually contains a number of procedures and each procedure might be calling zero, one, or more other procedures. Here, the entities are the procedures such that procedure P is connected to procedure P' if P calls P'; and the connections are referred as *inter-procedural calls*.

Evolving systems has properties like changeability [9][10], evolvability [11], and stability [12]. Changeability is system's ability to change over time such that its generation will not change. Stability is system's ability to endure change and evolution over time. We aim to compute the changeability of an evolving system.

The motivation is to *intelligently compute association rules* for an evolving system [13][14][15][16][17]. Traditionally, the interesting association rules has support and confidence exceeding given thresholds minimum support (minSup) and minimum confidence (minConf) [18][19]. Extensively, we intelligently compute the *stability* property of *network evolution rules* for an evolving system. For a given set of states $\mathbb{SS}$ of an evolving system, we are interested in mining rules of the form $X \rightarrow Y$, called Network Evolution Rules (NERs), with the following characteristics:

(a) the rule is "interesting", in the sense that if the source entities in set X exist in a state, then target entities in set Y will also present in that state.

(b) the rule is "stable", in the sense that its "stability" exceeds a given threshold, where "stability" is defined to be the percentage of states in $\mathbb{SS}$ in which the rule is interesting.

The first parameter ("interestingness") characterizes the behaviour of the NER within an individual state S_i of $\mathbb{SS}$, whereas the second parameter ("stability") characterizes the behaviour of the whole set of states $\mathbb{SS}$. The advantage of "stability" is to retrieve Stable NERs (SNERs) that remains stable (or persistent) in sufficient number of states over time.

We are also motivated to compute the system's changeability metric for system evolution analysis [20][21]. We use NERs and SNERs counts to compute the changeability for an evolving system. The proposed approach is useful for an evolving system whose states can be represented as *system network databases*. For a real-world evolving system, the challenge is to construct and mine a set of dynamic databases over the multiple states.

The main contributions of the paper are as follows. We introduce stability as a new measure for characterizing NERs over time, which helps to retrieve SNERs. We introduce a novel formula to compute system Changeability Metric (CM), which aids in quantitative system analysis. For this, we propose a new algorithm for SNERs Mining and Changeability Metric (SNERM_CM). We demonstrate experimental analysis by applying our intelligent approach for six real-world evolving systems.

Rest of the paper is organized as follows. Section II describes related state-of-the-arts. Section III describes the new definitions and changeability metric. Section IV presents the proposed SNERM_CM algorithm, which is further used to develop an intelligent tool. Section V describes the experiments by applying the tool on six evolving systems. Section VI describes conclusion.

II. Related Work

This section presents current state-of-the-arts that are comparable with our approach. This paper is inspired by recent advancements in the study of system evolution based on graph evolution mining. In the end, we will describe the possible applications of our work.

To begin with, contribution related to the computational intelligence based network rule mining includes following. Abonyi et al. [13] presented a comprehensive view about the links between computational intelligence and data mining; this is supported with a case study to extract knowledge represented by fuzzy rule-based expert systems that uses data mining algorithms. Liu et al. [14] presented intelligent computation of association rules based on a fixpoint operator for computing frequent itemsets. Ting et al. [15] present a study of linkage discovery (a topic in Genetic Algorithms) by using association rule mining instead of applying GAs for data mining. Extensively, we use *stable network evolution rule mining* algorithm on the *evolving computation intelligence systems* [16][17] represented as a set of evolving networks.

Liu et al. [18] presented a statistical approach to analyse temporal databases to identify stable rules, trend rules, and removed unstable rules. Extensively, we introduce an algorithm to retrieve Stable Network Evolution Rules (SNERs).

Contributions related to the graph evolution rules include following works. Berlingerio et al. [26] proposed an approach to mine Graph Evolution Rules (GERs) and shown four real world applications. After that, Leung et al. [27] proposed mining Link Formation Rules (LFRs) containing link patterns in social networks. Subsequently, Fan et al. [28] proposed Graph-Pattern Association Rules (GPARs) for entities in social graphs. Scharwächter et al. [29] proposed an EvoMine, a tool to mine frequent graph evolution rules with insertions and deletions of edges and nodes using labelling on node and edge. Additionally, they [29] also compared the GERM, LFR miner, with their EvoMine. The strength of our proposed SNERM over the current state-of-art is as following. First, in contrast to GERs [26], LFRs [27], GPARs [28], and EvoMine [29], we identify NERs and then compute stability of NERs to retrieve the SNERs. The existing mining algorithms do not use the minimum stability (minStab) measure, which is novel in our approach. Second, the existing mining algorithms are not extended to find system changeability metric. The stability of SNERs provides important statistical information about the NERs over time to compute the changeability metric.

Contributions related to the change mining include following works. Böttcher et al. [30][31] presented change mining and contrast mining. They defined change mining as data mining over a volatile and evolving repository with an objective of understanding evolution. They stated change analysis as; *contrast mining* if done on two data instances, and; *change mining* if done on multiple data instances. Change mining is one of the major problem domains of data mining, which have numerous applications. The datasets in which change mining is applied are as follows: process management systems [32], online data streams [33], retail marketing [34], real life application [35], patent trends [36], and changes in customer behaviors [37]. Takaffoli et al. [38] presented a framework to model and detect community evolution in social networks; a community-matching algorithm is used to identify and track similar communities over time. In contrast, we took advantage of graph (network) theories. Our approach is applicable to any kind of evolving system represented as a set of evolving system networks.

Contributions related to the complex system analysis includes following works. Liu et al. [39] developed analytical tool to study controllability of complex self-organized systems; the tool identifies a set of driver nodes with time-dependent control that can guide the system dynamics. Thereafter, Liu et al. [40] developed an approach to study observability by reconstructing the system's internal state from its outputs; the approach identifies sensors that are used to monitor a selected subset of state variables. In contrast, our approach and tool finds nodes in the antecedent and consequent of a rule; further, we use NERs and SNERs count to study changeability.

Contributions related to the changeability analysis includes following works. Ross and Rhodes [41] introduced Epoch-Era Analysis (EEA) as an approach to model a tradespace exploration process about the temporal system value environment. Thereafter, the EEA is used to identify changeability in system design [42] and sustaining lifecycle value of system [43]. The EEA is extended by Fitzgerald et al. [44] based on Valuation Approach for Strategic Changeability (VASC) to assess changeability over a system's lifecycle. As an application, Koh et al. [45] assess the changeability of complex engineering systems, and Fluri [46] assessed changeability of source code entities in the evolving software systems. Xuan et al. [47] proposed keyword association line network (KALN) to do uncertainty analysis of keyword system for web events. Recently, Avalos et al. [48] analyzed impact of changeability during external agent changes, and analyzed change affects the evolution on the complex adaptive system (CAS). In contrast to these contributions [41]-[48] for changeability analysis, we made following contribution and improvement. Our approach computes novel changeability metric for a state series of an evolving system. Our approach is based on knowledge discover in databases (KDD) and network (graph) theory. We also demonstrate its application on six evolving systems.

Our approach is useful for the application of evolving graph mining and temporal network analysis. Specially, where there is a need to identify stable (or persistent) rules of system entities over time. Our approach has many applications, which includes: temporal networks analysis of social media [49], analyzing sensor data using rule-based technique [50], and human interaction patterns in temporal networks [51].

Our approach can be applicable to analyze social network structures [52], to characterize communication network motifs [53], and to do systems thinking by aiding systems modeling language (SysML) [54]. Some other application areas of our work are as following: biological network, computer network, economical network, attackers-victims patterns in computer networks, star patterns in sky, co-author patterns over a decade in scholarly data, and family-members patterns over few generations in social sciences studies.

In the Section V, we demonstrate the two applications. First, for an evolving software system, which is an active working problem. Recently, Di Nucci et al. [55] proposed a technique to select a set of classifiers, which are better to predict the software bug proneness. We demonstrate our technique on a software repository of Hadoop-HDFS. Second, for evolving natural language processing, which is a current working issue in computation intelligence [56]. We demonstrate our technique over repositories of natural language system.

III. Stable Network Evolution Rules for System Changeability Metric

In this section, we present formal definitions and notations. This includes our key definitions of Network rule, Connection pairs, Network Evolution Rule (NER) and Stable Network Evolution Rule (SNER) with their support, confidence, and stability. At last, we formulate system's Changeability Metric.

As mentioned earlier, an evolving system contains distinct interconnected entities in an evolving system network. If entity e has connecting direction toward entity e', then we call e the *source* and e' the *target* of the connection. A convenient way for representation is directed graph to represent connections between entities in a given state S_i. We call this graph, *system network* of S_i such that nodes represent entities and edges represent connections (e→e' if and only if source entity e is connected to target entity e').

Given a state S_i of an evolving system, we shall denote the set of source entities by sS_i and the set of target entities by tS_i. Let $\mathbb{SS} = \{S_1, S_2 \ldots S_N\}$ be a set of states of an evolving system. Let $s\mathbb{SS}$ be the set of all source entities in $\mathbb{SS}$, that is, $s\mathbb{SS} = sS_1 \cup sS_2 \cup \ldots \cup sS_N$; and let $t\mathbb{SS} = tS_1 \cup tS_2 \cup \ldots \cup tS_N$ be the set of all target entities in $\mathbb{SS}$. Note that $s\mathbb{SS}$ and $t\mathbb{SS}$ are not necessarily disjoint, which means an entity can be a source and a target at the same time.

Definition 1: A <u>*Network Rule*</u> (NR) in state S_i is an expression of the form $X \rightarrow Y$, where X is a subset of source entities sS_i and Y is a subset of target entities tS_i.

A *network rule* can be interpreted as "if source entity (or entities) occurs in set X, then target entity (or entities) in Y are likely to occur with a given support and confidence in a connection pair". To define the support and confidence of a network rule in a state S_i, we first define the concept of "connection pair" in S_i.

Definition 2: A <u>*connection pair*</u> in S_i is an ordered pair (L, R) such that L is a subset of source entities and R is a subset of target entities. For each e in L there exit e' in R that makes a connection e→e' in the network of state S_i. Where, the symbols L and R stand for "left" and "right".

The support and confidence of a network rule in a state S_i is computed with respect to a given set of connection pairs. Such a set is defined with respect to some aspect of the evolving system. However, we shall assume that a set of connection pairs is given. Different kind of systems has different mechanism to generate their system network. Thus, mechanism to collect connection pairs depends upon pre-processing of system network. For example, if an evolving system is a software system then it is usually structured as modules and each module contains a number of procedures (as entities). In each module, each procedure calls zero, one, or more procedures in the same module or in different modules. Therefore, each module determines a connection pair between its own set of procedures, say L, and the set of target procedures, say R.

The connection pairs of a state create a *System Network Database* (*SysNetDb*). A collection of system network databases for multiple states is referred as *SysNetDb<u>s</u>*. An example for the list of multi-sport events database is presented in Fig. 1, which has one connection pair in first two SysNetDbs and 3 connection pairs in third SysNetDb. By pre-processing, we eliminated stop-words and kept only keywords (as entities).

We shall say that a set W of entities occurs in a connection pair (L, R) if W is a subset of $L \cup R$. During pre-processing, each connection pair (L, R) is transformed into a numerical sequence based on the following two steps:

- The set of all entities appearing in state S_i are encoded as positive integers, that is, each entity is associated with a unique positive integer. For example, the Index in Fig. 1 has 13 entities, which are encoded to transform the SysNetDb.
- For each connection pair (L, R), we replace the entities in L and in R with the sets of their encodings. The encodings of all connection pairs are stored in a database, which we represent them as SysNetDb_i. For example, in Fig. 1 the SysNetDb_189, SysNetDb_190, and SysNetDb_191 denotes SysNetDbs for decades (as states) 1890, 1900, and 1910 respectively.

Definition 3: Given a set of connection pairs in S_i, let $X \rightarrow Y$ be a network rule and let W be a set of entities in $X \cup Y$.

- The <u>*support count*</u> of W in S_i is denoted as supCount(W, S_i), and defined by supCount(W, S_i) = m, where m is the number of connection pairs in which W occurs.
- The <u>*support*</u> of W in S_i is denoted as sup(W, S_i), and defined by sup(W, S_i) = supCount(W, S_i) ÷ card(S_i), where card(S_i) is the cardinality (number) of connection pairs in state S_i.
- The <u>*support*</u> of $X \rightarrow Y$ in S_i is denoted as sup($X \rightarrow Y$, S_i), and defined by: sup($X \rightarrow Y$, S_i) = sup($X \cup Y$, S_i) = sup(W, S_i).
- The <u>*confidence*</u> of $X \rightarrow Y$ in S_i is denoted as conf($X \rightarrow Y$, S_i), and defined by: conf($X \rightarrow Y$, S_i) = sup($X \cup Y$, S_i) ÷ sup(X, S_i)
- An <u>*interesting network rule*</u> has support (or support count) and confidence greater than thresholds minSup (or minSupCount) and minConf, respectively.

Fig. 1 also shows an illustration for Network Rule Mining (NRM). Use "input of NRM" along with threshold minSupCount = 1 and minConf = 0.5 to generate "output of NRM" containing network rules. The support count is convertible to the support percentage by dividing supCount with the number of connection pairs (i.e. 3). Thus, minSup is equal to minSupCount (= 1) divided by cardinality (= 3) for SysNetDb_191, which is equal to 1/3. In Fig. 1, the network rule in "output of NRM" is in the order of their interestingness i.e. top most rule (in first row) is highest interesting, rules in middle row are moderately interesting in decreasing order, and last rule is least interesting.

Definition 4: A *Network Evolution Rule* (NER) in $\mathbb{SS}$ is an expression of the form X→Y, where X is a subset of source entities s$\mathbb{SS}$ and Y is a subset of target entities t$\mathbb{SS}$. Each NER has a characteristic *stability* because it is distinct and interesting in one or more states.

- The *stability count* of X→Y in $\mathbb{SS}$ is denoted as stabCount(X→Y, $\mathbb{SS}$), and defined by stabCount(X→Y, $\mathbb{SS}$) = n, where n is the number of states in $\mathbb{SS}$ that has X→Y as interesting NER.

- The *stability* of X→Y in $\mathbb{SS}$ is denoted as stab(X→Y, $\mathbb{SS}$), and defined by stab(X→Y, $\mathbb{SS}$) = stabCount(X→Y, $\mathbb{SS}$) ÷ card($\mathbb{SS}$), where card($\mathbb{SS}$) is the cardinality of $\mathbb{SS}$ i.e. number of states N in $\mathbb{SS}$.

- The NER X→Y is defined to be *stable* in $\mathbb{SS}$ if its stability (or stability count) is greater minStab (or minStabCount); such NERs are *Stable Network Evolution Rules* (SNERs).

The stability is a new measure for characterizing NERs over time. The stability of a NER is our first conceptual contribution. The stability count of a NER (stabCount) is the frequency of states in which the NER is interesting. Consequently, *stability of NER* is the percentage of states in which the NER is interesting. The SNER is a new information for system network evolution over time and constitute our second conceptual contributions. The functional meaning of the discovered NER is frequent network rule of interconnected entities occurring in a SysNetDb of a state. The functional meaning of a SNER is the frequent NER of inter-connected entities occurring in SysNetDbs for a set of states.

Next, we use the numerical values of input-output parameters in the SNER mining to do the system changeability analysis. The changeability is a system property, which depends upon both evolution and stability information of an evolving system. To compute the changeability (as third conceptual contribution) for an execution of SNER mining, we defined following metric.

Definition 5: A *Changeability Metric* (CM) in a set of states $\mathbb{SS}$ of an evolving system is defined as

$$\mathrm{CM} = \frac{\mathrm{N}}{\mathrm{minStabCount}} * \frac{\mathrm{NER\_Count}}{\mathrm{SNER\_Count}} * 100$$

where the N is the number of states and the minStabCount is the threshold value used for SNER mining. Where, the NER_Count is the number of NERs retrieved and the SNER_Count is the number of SNERs retrieved. Note, the value of (N/minStabCount) is equal to (1/minStab) i.e. inverse of stability (in Definition 4).

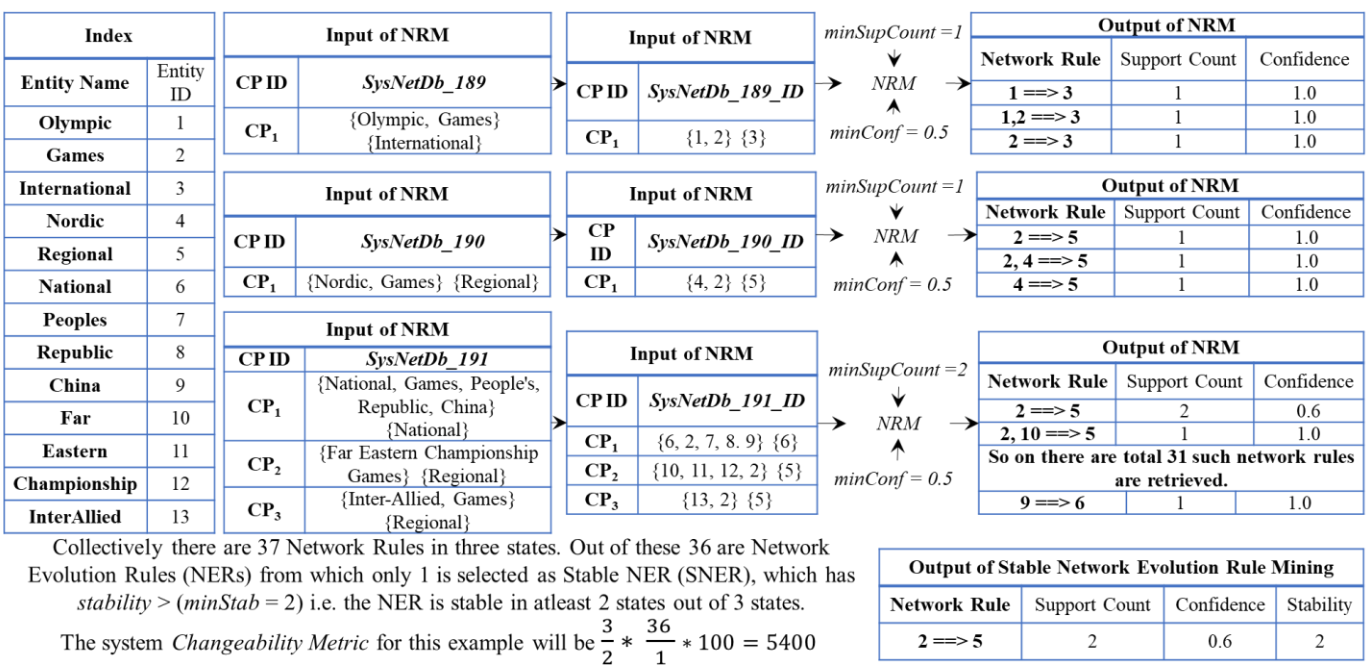


Fig 1. An illustrative example of the proposed approach to mine NERs, SNERs, and Changeability metric.

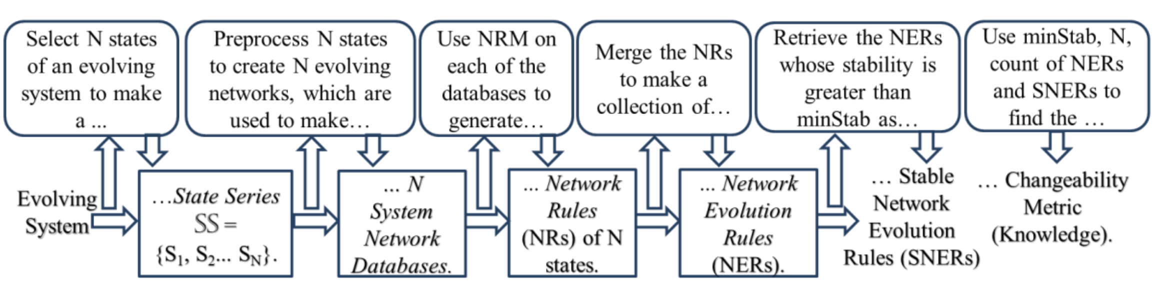


Fig 2. The process-artifacts involved to intelligently compute the SNERs and the Changeability Metric. Where, each rounded-rectangle shows a process and rectangle shows input-output artifact as per arrows. Each sentence starts from process at top and finishes at the artifact box.

In Fig. 1, the SNER mining resulted in 36 NER and 1 SNER, whose execution time was 1 second. This resulted in 5400 as CM value. Fig. 2 describes the summary of all the steps to intelligently compute the SNER and the Changeability metric. These steps are according to the guideline steps of Knowledge Discovery in Databases (KDD). The advantage of the new measure for *stability of NERs* is to compute *changeability* property of an evolving system. Next section presents an algorithm for mining SNERs from a given set of states $\mathbb{SS}$.

IV. SNERM AND CHANGEABILITY METRIC ALGORITHM

This section describes the algorithm SNERM_CM (Stable Network Evolution Rule Mining and Changeability Metric), which addresses two challenges. First, SNERM_CM uses a series of System Network Databases (SysNetDbs) representing a set of evolving system's states. Second, SNERM_CM uses network rule mining to perform mining over the SysNetDbs.

Before using SNERM_CM, we pre-processed a set of states $\mathbb{SS}$ to make a set of networks that are further used to make SysNetDbs. We use these databases in the SNERM_CM along with following input and output.

- inputs: SysNetDbs of $\mathbb{SS}$, minimum support count, minimum confidence, and minimum stability count.
- outputs: Network Evolution Rules (NERs) and Stable Network Evolution Rules (SNERs).

Firstly, the SNERM_CM algorithm identifies Network Rules (NRs) using a Network Rule Mining (NRM) algorithm. The **NRM** algorithm depends upon the sequential rule mining over the set of connection pairs (in SysNetDbs). The source and target sets in a connection pair is a sequence of a sequence database. A network rule (as a sequential rule) can be identified using minimum support and mining confidence on the network database (as a sequential database). Thus, intuitively **NRM** is reducible to **SRM** because the SRM efficiently used as a subroutine to solve the NRM efficiently. This reduction of transforming NRM algorithm to the SRM algorithm is our fourth conceptual contribution. We accomplished this by applying the **NRM** algorithm on *SysNetDb_i* to identify the network rules (*net_Rules_i*) for a state S_i. A network rule X→Y has antecedent X as a subset of source entities and consequent Y as a subset of target entities for connection pairs.

Secondly, using **Merge** algorithm the network rules of different states are merged together to make a collection of network rules (*Collect_NRs*). From this collection, we select unique (or distinct) network rules over states as NERs.

Thirdly, for each NER, we count its number of distinct occurrence in states - the count is the 'stability count' of the NER. An expert specifies a threshold minStabCount, which aids to retrieve SNERs from NERs whose stability count is greater than minStabCount value.

The retrieved NERs and SNERs are generated as the output. In the output, the SNERs are more meaningful as compared to NERs because a SNER is interesting as well as stable in an evolving system. Here, 'meaningful' means selected SNERs have more 'stability' than the unselected NERs, where stability is measured with the threshold minStab

The meaningful NERs depend upon the threshold minStab. If the minStab is higher than the expected range of meaningful NERs, then less number of most meaningful SNERs are

---

Algorithm **SNERM_CM (***SysNetDbs*, *minSupCount*, *minConf*, *minStabCount***)**

Initialize String *net_Rules_i*
Initialize Array *NRs*, *Collect_NRs*, *SNER*
Initialize HashMap *NERs_HM* < *NER*, *stability* >
Initialize File *NERs*, *SNERs*
Initialize *i* ∈ integer
Initialize *CM* ∈ float
For each state *SysNetDb_i* in *SysNetDbs*
  net_Rules_i = **NRM(***SysNetDb_i*, *minSupCount*, *minConf***)**
  Store a new file *net_Rules_i* in directory *netRules*
End For
For each state *net_Rules_i* in *netRules*
  Collect_NRs = **Merge(***Collect_NRs*, *net_Rules_i***)**
End For
For each distinct rule (as *NER*) in *Collect_NRs*
  Initialize int *stabilityCount* = 0
  For each *rule_x* in *Collect_NRs*
    if (*NER* is identical to *rule_x***)**
      then *stabilityCount++*
    end if
  End for
  if (*NER* is not in *NERs_HM***)**
    then Add (<*NER*, *stabilityCount*> to *NERs_HM***)**
  end if
  if (*NER* is not in *NERs***)**
    then Add (*NER* to *NERs***)**
  end if
  if (*stabilityCount* > *minStabCount***)**
    if (*NER* is not in *SNERs***)**
      then Add (*NER* to *SNERs***)**
    end if
  end if
End For
Find NERs_Count and SNERs_Count, then use Changeability Metric formula to compute and store in *CM*
Return *NERs*, *SNERs*, *CM*

Algorithm **NRM(***SysNetDb_i*, *minSupCount*, *minConf***)**

seq_rule_i = **SRM (***SysNetDb_i*, *minSupCount*, *minConf***)**
// The **NRM** is reducible to the sequential rule mining (**SRM**) because SysNetDb is a kind of sequence database, where each connection pair is a sequence of source and target sets.
// The SRM uses *minSupCount* and *minConf* to generate sequential rules (i.e. network rules with source and target sets.
Return *net_Rules_i*

Algorithm **Merge(***Collect_NRs*, *net_Rules_i***)**

For each distinct network rule (*NR*) in *net_Rules_i*
  append(*NR***) to** *Collect_NRs*
// This makes multiple *NR* for multiple states.
// These *NR* in *Collect_NRs* is referred as *NER*.
// This multiple entries of *NR* helps to calculate stability of the NER in the SNERM algorithm.
Return *Collect_NRs*

retrieved. Conversely, if the minStab is lower than the expected range of meaningful NERs, then the retrieved SNERs includes both meaningful and meaningless NERs. On one hand, for a high value of minStab, there might be no SNERs. On the other hand, for a low value of minStab, there might be exhaustive number of SNERs.

The NERs are not retrieved as SNERs belongs to the set of less stable rules, which can be considered as interesting but unstable NERs for a threshold. Such NERs also have support, confidence, and stability; however such NERs has stability lesser than the minStab. The thresholds are measured by the values of minSupCount-minConf–minStabCount that are decided by performing experiments.

The SNERM_CM finds and compares the counts of NERs and SNERs. Thereafter, the algorithm uses changeability metric formula to compute the system changeability metric (CM). The computational complexity of the SNERM_CM algorithm mainly depends upon $O(N\times\lambda)$, where λ is the complexity of NRM algorithm.

Based on our approach, we developed a prototype tool on Java. Next section discusses experimental results by applying our tool for six real-world evolving systems.

V. Experimentation and Results

We used our tool to show the practical implication of our approach and algorithm described in Section III and Section IV. Our tool internally uses fundamental Sequential Rule Mining (SRM) [22] (a kind of association rule mining [19]) to do Network Rule Mining (NRM). Specifically, for each state, our tool used RuleGrowth [23][24][25] to generate network rules using a set of connection pairs in a SysNetDb. Rest of the approach is same as described in the SNERM_CM algorithm.

Using our tool, we conducted experimentation on six evolving systems shown in Table I, where the first column is list of evolving system names. The second column listed the number of states used in the experimentation for an evolving system. The third column listed the number of entities in an evolving system. The fourth column listed the experimental

TABLE I
Information about SNERM experiments conducted on the six evolving systems.

Evolving Systems	N	Number of entities	minSupCount-minConf-minStabCount	NER Count	Stable NER Count	Number of source and target entities	Changeability metric
HDFS-Core	15	3129	4-0.4-3	5	4	4 & 4	6.25
List of Bible Translation	13	246	3-0.3-2	3715	16	2 & 4	1509.21
List of Multi-sport Events	13	141	3-0.2-2	11	6	3 & 2	11.91
Retail Market	13	1872	4-0.6-3	131	12	12 & 2	46.94
Positive sentiment of movie genres	16	284	2-0.3-4	146	5	5 & 3	116.8
Negative sentiment of movie genres	16	510	2-0.3-4	258	10	9 & 5	103.2

TABLE II
For each evolving system, the Stable NERs are generated based on the minSupCount-minConf-minStabCount as mentioned in Table I.

Hadoop-HDFS[1]	List of Multi-sport Events[3]	Positive[6] sentiment in IMDb[5]
create ==> convert readAll ==> readFully checkAccess ==> getDelegationToken close ==> getRemoteAddressString	World ==> International Games ==> Regional Games ==> International Games, World ==> International Games, Asian ==> Regional Asian ==> Regional	premier ==> Short fond ==> Short fine ==> Short humor ==> Comedy grand ==> Music
List of Bible Translation[2]	**Retail Market[4]**	**Neagative[6] sentiment in IMDb[5]**
English ==> Vulgate English ==> Masoretic English, Modern ==> Masoretic English ==> Masoretic, Text English, Modern ==> Masoretic, Text English ==> Text English, Modern ==> Text English ==> Text, Greek English, Modern ==> Text, Greek English ==> Greek English, Modern ==> Greek Modern ==> Masoretic Modern ==> Masoretic, Text Modern ==> Text Modern ==> Text, Greek Modern ==> Greek	SUKI SHOULDER BAG ==> 17841 ASSORTED MONKEY SUCTION CUP HOOK ==> 17841 CARRIAGE ==> 14911 SKULL DESIGN TV DINNER TRAY ==> 17841 ASSORTED COLOUR LIZARD SUCTION HOOK ==> 17841 SMALL YELLOW BABUSHKA NOTEBOOK ==> 17841 LIPSTICK PEN RED ==> 17841 UNION STRIPE CUSHION COVER ==> 17841 DISCO BALL CHRISTMAS DECORATION ==> 17841 BLUE/CREAM STRIPE CUSHION COVER ==> 17841 CAKE PLATE LOVEBIRD WHITE ==> 17841 KITCHEN METAL SIGN ==> 17841	rue ==> Short terrible ==> Short rue ==> Documentary pain ==> Short vent ==> Short passe ==> Short sin ==> Drama brat ==> Drama terror ==> Horror perverse ==> Adult

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

thresholds of minSupCount-minConf-minStabCount. Here, minSupCount and minStabCount are in frequency, which are convertible to percentage minSup and minStab. Where, the minSupCount is a threshold for the number of entities occurring together in the connection pairs, and the minStabCount is a threshold for the number of states in which NER occurs.

As we used fixed size of individual system network database for each evolving systems' experiments, thus both support count and support percentage have same significance. Hence, we used minSupCount for thresholding the support count (since integer value is simple to interpret as compared to float). Similarly, we used fixed number of states (N) for each evolving systems' experiments, thus both stability count and stability percentage have same significance. Hence, we used minStabCount for thresholding the stability count.

Table I also demonstrates the empirical evaluation of the SNERM_CM experiment by mentioning the experimental results in following columns. The fifth column provides the number of NERs retrieved in the experiments. The sixth column provides number of SNERs retrieved. The seventh column provides the number of distinct source and target entities in the SNERs. The eighth column provides changeability metric value computed from the given details about experiment in each row.

A system data engineer wants to report less number of manageable and meaningful rules by choosing best-possible high value of thresholds. The meaningfulness is a qualitative measure and optimized with the quantitative values of the threshold. Although we can generate large number of NERs, such exhaustive number of NERs, which have both meaningful and meaningless NERs. Thus, we used explore and exploit theory to choose of threshold values: minSupCount, minConf, and minStabCount. We explored minSupCount and minConf to generate optimized number of NERs, and then exploited the minStab to generate meaningful SNERs. Experiments are done for the best-possible high values of thresholds to generate manageable number of NERs and meaningful SNERs. Such NERs and SNERs are helpful in inferencing, decision-making, and action-taking.

Table II shows the SNERs retrieved for the experiment mentioned in Table I. The experiments are done for the thresholds given in the fourth column of Table I. We show only SNERs because they are interesting as well as stable. The NERs and SNERs are generated using SNERM_CM code for a set of evolving system's states. The SNERs (in Table II) represents antecedents as set of source entities and consequents as set of target entities. To do experiments, we used *set of states* as follows: set *of versions* for the Hadoop-HDFS (a software), *set of centuries* for the list of bible translation, *set of decades* for the list of multi-sport events, *set of months* for the retail market data, *set of decades* for the Internet Movie Database (IMDb).

We make inferences from the SNERs that are mentioned in Table II. We also explain the advantages of those SNERs for their evolving systems. The inferences and advantages of the SNERs for the six evolving systems are as follows.

A. Hadoop-HDFS

In Hadoop-HDFS, the SNER "readAll ==> readFully" means that if procedure 'realAll' is present in a module then most probably 'readFully' is also present in that module. Similarly, the other rules can also be interpreted. The advantage of these SNERs is in program analysis to do efficient software maintenance and evolution automatically.

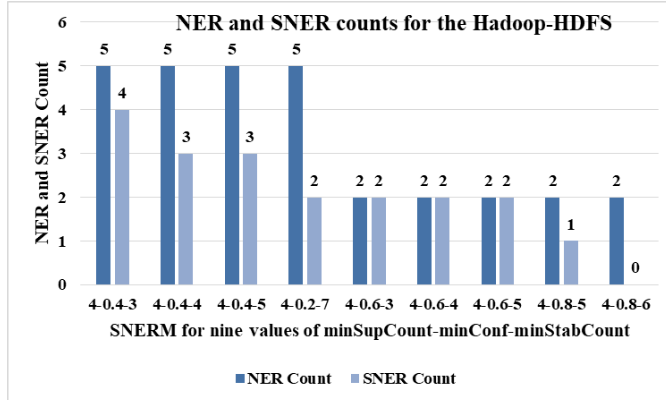


Fig 3. SNERM experiments for the Hadoop-HDFS.

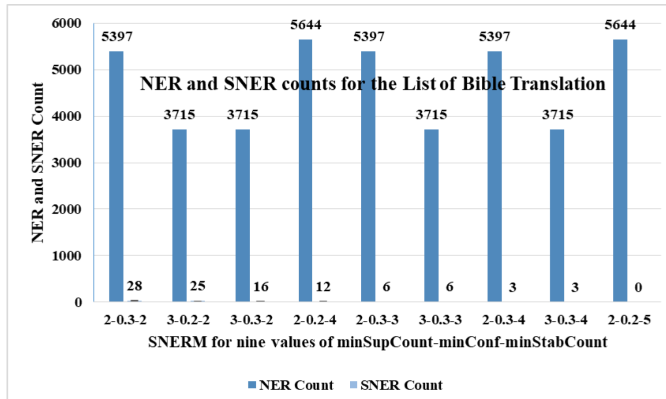


Fig 4. SNERM experiments for the list of bible translation.

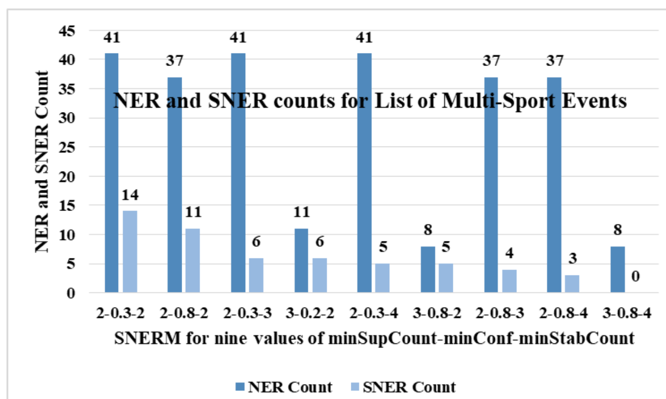


Fig 5. SNERM experiments for the list of multi-sport events system.

B. List of Bible Translation

In List of Bible Translation, the SNERs suggest that most of the 'Modern English' bibles are translated from the 'Valgute', 'Masoretic Text', and 'Greek Text'. The advantage of these SNERs is in natural language processing to study bible evolution automatically.

C. List of Multi-sport Events

In the List of Multi-sport Events, the SNERs suggest that events named with 'World Games' are probably having 'International' scope (level) and events named with 'Asian Games' are probably having 'Regional' scope (level). The advantage of these SNERs is in natural language processing to study multi-sport event evolution automatically.

D. *Retail Market*

In Retail Market, the SNER suggest that the customer with ID ‘17841’ frequently purchases shopping items listed as the antecedent of the rules. Similarly, the customer with ID ‘14911’ frequently purchases ‘carriage’ as the shopping item. These shopping items are useful to do target marketing on such customers. The advantage of these SNERs is in automated market analysis.

E. *Positive sentiment in IMDb*

In the positive sentiment based dataset of IMDb, a rule suggest that movie name with positive sentiment words like ‘premier’, ‘fond’, and ‘fine’ probably belongs to ‘Short’ movie genre. Similarly, positive sentiment word ‘humor’ most probably appears in movies of genre: Comedy. Similarly, positive sentiment word ‘grand’ most probably appears in movies of genre: Music. The advantage of these SNERs is in automated positive sentiment analysis of movie names-genre.

F. *Negative sentiment in IMDb*

In the negative sentiment of IMDb result, a rule suggest that movie name with positive sentiment words like ‘rue’, ‘terrible’, ‘pain’, ‘vent’, ‘passe’ probably belongs to ‘Short’ movie genre. Similarly, negative sentiment word ‘rue’ most probably appears in movies of genre: Documentary. Similarly, negative sentiment words ‘sin’ and ‘brat’ most probably appears in movies of genre: Drama. Similarly, ‘terror’ and ‘perverse’ belongs to movie genres Horror and Adult respectively. The advantage of these SNERs is in automated negative sentiment analysis of movie names-genre.

Interpreting NER and SNER count optimization in Figs. 3, 4, 5, 6, 7, and 8, which describes a brief overview to demonstrate experimentation results. Each figure is a bar chart that demonstrates nine pair of bars to represent nine experiments. In each bar chart, horizontal-axis depicts the values of *minSupCount-minConf-minStabCount* for which experiments are performed to generate NERs and SNERs. In each bar chart, vertical-axis depicts the count of NERs or SNERs; such that the SNERs count is in decreasing order. Each pair of bars represents the number of NERs and SNERs retrieved for single execution of mining NERs and SNERs algorithm. Each figure shows nine pair of bars for nine such executions. Each pair of bars is for a combination of the values of minSupCount-minConf-minStabCount used in SNERM_CM algorithm.

From Figs. 3, 4, 5, 6, 7, and 8, we can observe following three inferences for relative changeability between six evolving systems.

- In Figs. 3 and 5 of the Hadoop-HDFS and the List of Multi-Sport Events, respectively - for an experiment - the difference between the counts of SNERs and NERs are less. Thus, both of the systems are less prone to changeability.
- The evolving systems of Figs. 6, 7, and 8 have moderate difference between the counts of SNERs and NERs (for an experiment). Thus, the three evolving systems are moderately prone to changeability.
- In Fig. 4 of the List of Bible Translation, the difference between the counts of SNERs and NERs are high. Thus, it is highly prone to changeability.

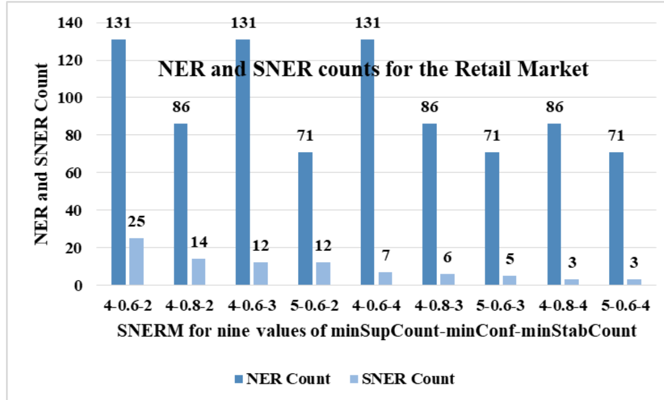


Fig 6. SNERM experiments for the retail market.

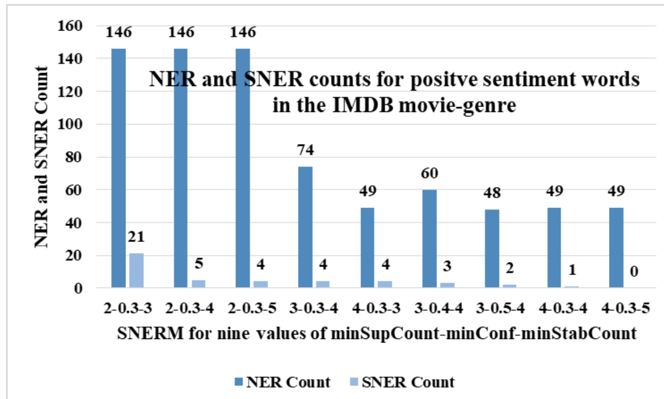


Fig 7. SNERM experiments for the positive sentiment in IMDb data.

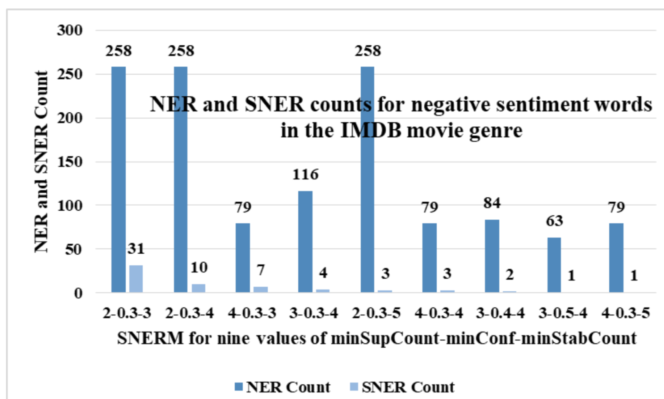


Fig 8. SNERM experiments for the negative sentiment in IMDb data.

The above three inferences are also true in case of the changeability metric value given for the six evolving systems in Table 1. The NERs and SNERs can also be retrievable in other fields like bioinformatics data, geological data, agricultural data etc. Such domains can use and take advantage of the SNERM_CM algorithm. The SNERM_CM applied on an evolving system can help a subject matter specialist (or domain expert).

Although we presented novel technique to retrieve NERs, but still used NERs as baseline for SNERs to show improvement. Both NER and SNER are “interesting” (same as an interesting association rule), but a SNER is also “stable” in the system states. If we assume the retrieved NERs as the baseline then the SNERs are the improvements. We compared this improvement in terms of the number of stable rules (SNERs) with respect to the remaining unstable rules (in NERs). The figures 3, 4, 5, 6, 7, and 8 shows pair of bars as improved optimization such that the number of NERs is larger than number of SNERs. This provides the quantitative improvement in efficiency due to retrieved SNERs form NERs.

In addition to the advantages of SNERs as compared to the baseline NERs, we also compared our approach with the existing state-of-the-arts (in Section II). Despite unavailable exact comparable experiments; nevertheless, in the Section II we compared of our approach with some similar existing approaches, and described applications of our approach.

The accuracy of the NERs and SNERs depends upon the algorithm for Network Rule Mining (NRM), which further depends upon the Sequential Rule Mining (SRM). The execution time of SNERM_CM algorithm mainly depends upon four factors. First factor is the chosen SRM algorithm. Second, the database size used for the SysNetDbs; larger the database size more the execution time and smaller the database size lesser the execution time. Third factor is the threshold values used for experiments; lower values need more execution time and higher values need less execution time. Fourth factor is the complex connections between the entities used to make connection pairs; complex network connections make a complex SysNetDb and simpler network connections make simpler SysNetDb. The complexity of network connections to create SysNetDb depends upon system domain of the dataset. Higher the complexity of network to create SysNetDb results in higher execution time. Lower the complexity of network to create SysNetDb results in lesser execution time.

VI. CONCLUSION

In this paper, we introduced stability as a new measure for characterizing Network Evolution Rules (NERs) over time. We did this using a proposed algorithm SNER Mining and Changeability Metric (SNERM_CM), which presented an approach to retrieve NERs and Stable NERs (SNERs) in a set of states $\mathbb{SS} = \{S_1, S_2 \ldots S_N\}$ of an evolving system. The purpose of SNERs is to study and compute the changeability property of an evolving system.

Based on SNERM_CM algorithm, we developed an intelligent tool, which is used to do experiments on evolving systems. We also demonstrated applications of the tool on six evolving systems. We summarized experimental results to show various SNERs for some chosen value of minimum support count (minSupCount), minimum confidence (minConf), and minimum stability count (minStabCount). We also demonstrated the optimization in the number of SNERs as compared to NERs. To the best of our knowledge, we found our approach is novel in context of system evolution analysis based on SNERs and Changeability Metric.

We also used the six evolving systems to perform system evolution analytics [20][21] to calculate system network complexity [57] and to do system evolution recommendations [58]. In future, we will study the stability for an evolving system using these SNERs. Additionally, we will also present enhanced experimentations on the six evolving systems. We also planned to demonstrate few more applications of our tool for other evolving systems.

REFERENCES

[1] T. P. Hughes. "The evolution of large technological systems." The social construction of technological systems: New directions in the sociology and history of technology 82 (1987).

[2] J. H. Holland. "Complex adaptive systems." *Daedalus* (1992): 17-30.

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

[4] R. Valerdi, A. M. Ross, and D. H. Rhodes. "A framework for evolving system of systems engineering." *CrossTalk*. 2007.

[5] S. A. Frost, M. J. Balas. "Evolving Systems and Adaptive Key Component Control". In: Arif, T.T. (ed.) Aerospace Technologies Advancements. ISBN: 978-953-7619-96-1 (2010).

[6] C. Aggarwal, and S. Karthik. "Evolutionary network analysis: A survey." *ACM Computing Surveys (CSUR)* 47.1 (2014): 10.

[7] G. Ditzler, M. Roveri, C. Alippi, & R. Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10.4 (2015), 12-25.

[8] T. Mens, A. Serebrenik, and A. Cleve. Evolving Software Systems. *Springer Publishing Company, Incorporated*. 2014.

[9] E. Fricke, and A. P. Schulz. "Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle." *Systems Engineering* 8.4 (2005), 342-359.

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

[11] H. McManus, and D. Hastings. "3.4. 1 A Framework for Understanding Uncertainty and its Mitigation and Exploitation in Complex Systems." *INCOSE International Symposium*. 15.1. (2005), 484-503.

[12] M. W. Maier. "Architecting principles for systems-of-systems." *Systems engineering* 1.4 (1998), 267-284.

[13] J. Abonyi, B. Feil, and A. Abraham. "Computational intelligence in data mining." *Informatica* 29.1 (2005), 3-12.

[14] H. Liu & J. Zeleznikow "Intelligent computation for association rule mining". *In Workshop on Computational Intelligence in Data Mining in conjunction with the 5th IEEE International Conference on Data Mining* 2005, 49-53.

[15] C.-K. Ting, W.-M. Zeng, and T.-C. Lin. "Linkage discovery through data mining [research frontier]." *IEEE Computational Intelligence Magazine* 5.1 (2010): 10-13.

[16] P. Angelov, D. P. Filev, and N. Kasabov, eds. *Evolving intelligent systems: methodology and applications*. Vol. 12. John Wiley & Sons, 2010.

[17] P. Angelov, and N. Kasabov. "Evolving computational intelligence systems." *Proceedings of the 1st International Workshop on Genetic Fuzzy Systems*. 2005, 76-82.

[18] B. Liu, Y. Ma, and R. Lee. "Analyzing the interestingness of association rules from the temporal dimension." *IEEE International Conference on Data Mining*. ICDM. IEEE, 2001, 377-384.

[19] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," *ACM SIGMOD Conf. Management of Data*, May 1993, 207-216.

[20] A. Chaturvedi and A. Tiwari. "System Evolution Analytics: Evolution and Change Pattern Mining of Inter-Connected Entities". *IEEE International Conference on Systems, Man, and Cybernetics (SMC)* 2018, 3877-3882.

[21] A. Chaturvedi and A. Tiwari. "System Evolution Analytics: Deep Evolution and Change Learning of Inter-Connected Entities". *IEEE International Conference on Systems, Man, and Cybernetics (SMC)* 2018, 3075-3080.

[22] R. Agrawal, and R. Srikant. "Mining sequential patterns." *11th International Conference on Data Engineering*, IEEE, 1995, 3-14.

[23] P. Fournier-Viger, C. W. Wu, V. S. Tseng, L. Cao, & R. Nkambou. "Mining Partially-Ordered Sequential Rules Common to Multiple Sequences" *IEEE Transactions on Knowledge and Data Engineering*, 27.8, 2015, 2203-2216.

[24] P. Fournier-Viger, R. Nkambou, and V. S. Tseng. "RuleGrowth: mining sequential rules common to several sequences by pattern-growth." *ACM Symposium on Applied Computing,* 2011, 956-961.

[25] P. Fournier-Viger, et al. "SPMF: a Java open-source pattern mining library." *Journal of Machine Learning Research* 15.1 (2014): 3389-3393.

[26] M. Berlingerio, et al. "Mining graph evolution rules." *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2009, 115-130.

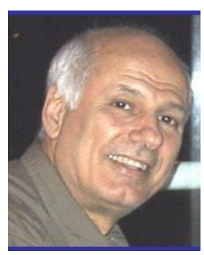
[27] C. W. Leung, et al. "Mining interesting link formation rules in social networks." *19th ACM International Conference on Information and Knowledge Management.* ACM, 2010, 209-218.

[28] W. Fan, et al. "Association rules with graph patterns." *VLDB Endowment* 8.12 (2015): 1502-1513.

[29] E. Scharwächter, et al. "Detecting Change Processes in Dynamic Networks by Frequent Graph Evolution Rule Mining." *IEEE 16th International Conference on Data Mining (ICDM).* IEEE, 2016, 1191-1196.
[30] M. Böttcher, F. Höppner, and M. Spiliopoulou. "On exploiting the power of time in data mining." ACM SIGKDD Explorations Newsletter 10.2 (2008): 3-11.
[31] M. Boettcher. "Contrast and change mining." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1.3 (2011): 215-230.
[32] C. W. Günther, et al. "Change mining in adaptive process management systems." On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE. Springer Berlin Heidelberg, 2006 309-326.
[33] G. Dong, et al. "Online mining of changes from data streams: Research problems and preliminary results." *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams.* 2003, 739-747.
[34] M.-C. Chen, A.-L. Chiu, and H.-H. Chang. "Mining changes in customer behavior in retail marketing." *Expert Systems with Applications* 28.4 (2005): 773-781
[35] B. Liu, et al. "Mining changes for real-life applications." *DaWaK.* Vol. 1874. 2000, 337-346.
[36] M.-J. Shih, D.-R. Liu, and M.-L. Hsu. "Discovering competitive intelligence by mining changes in patent trends." *Expert Systems with Applications* 37.4 (2010): 2882-2890.
[37] H. S. Song, J. k. Kim, and S. H. Kim. "Mining the change of customer behavior in an internet shopping mall." *Expert Systems with Applications* 21.3 (2001): 157-168.
[38] M. Takaffoli, et al. "Community evolution mining in dynamic social networks." *Procedia-Social and Behavioral Sciences* 22 (2011): 49-58.
[39] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási. "Controllability of complex networks." *Nature* 473.7346 (2011): 167.
[40] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási. "Observability of complex systems." *Proceedings of the National Academy of Sciences* 110.7 (2013): 2460-2465.
[41] A. M. Ross, and D. H. Rhodes. "11.1. 1 Using Natural Value-Centric Time Scales for Conceptualizing System Timelines through Epoch-Era Analysis." *INCOSE International Symposium*. 2008, 18.1, 1186-1201.
[42] M. E. Fitzgerald, A. M. Ross, and D. H. Rhodes. *A method using epoch-era analysis to identify valuable changeability in system design*. MASSACHUSETTS INST OF TECH CAMBRIDGE, 2011.
[43] M. E. Fitzgerald, and A. M. Ross. "Sustaining lifecycle value: Valuable changeability analysis with era simulation." *IEEE International Systems Conference (SysCon)*. IEEE, 2012, 1-7.
[44] M. E. Fitzgerald, A. M. Ross, and D. H. Rhodes. "8.4. 1 Assessing Uncertain Benefits: a Valuation Approach for Strategic Changeability (VASC)." *INCOSE International Symposium*. 22.1. 2012, 1147-1164.
[45] E. CY Koh, N. HM Caldwell, and P. J. Clarkson. "A technique to assess the changeability of complex engineering systems." *Journal of Engineering Design* 24.7 (2013): 477-498.
[46] B. Fluri. "Assessing changeability by investigating the propagation of change types." *Companion to the proceedings of the 29th International Conference on Software Engineering*. IEEE Computer Society, 2007, 97-98.
[47] J. Xuan, X. Luo, G. Zhang, J. Lu, and Z. Xu. "Uncertainty analysis for the keyword system of web events." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46, no. 6 (2016): 829-842.
[48] J. Avalos, M. W. Grenn, and B. Roberts. "Assessment of Complex Adaptive System Changeability Using a Learning Classifier System." *IEEE Systems Journal* (2018).
[49] P. Holme. "Analyzing temporal networks in social media." *Proceedings of the IEEE* 102.12 (2014): 1922-1933.
[50] F. Ganz, P. Barnaghi, and F. Carrez. "Automated semantic knowledge acquisition from sensor data." *IEEE Systems Journal* 10.3 (2016): 1214-1225.
[51] Y.-Q. Zhang, et al. "Human interactive patterns in temporal networks." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.2 (2015): 214-222.
[52] E. M. Jin, M. Girvan, and M. EJ Newman. "Structure of growing social networks." *Physical review E* 64.4 (2001): 046132.
[53] Q. Zhao, et al. "Communication motifs: a tool to characterize social communications." *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, 2010, 1645-1648.
[54] R. Cloutier, B. Sauser, M. Bone, and A. Taylor. "Transitioning systems thinking to model-based systems engineering: systemigrams to SysML models." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, no. 4 (2015): 662-674.
[55] D. Di Nucci, et al. "Dynamic selection of classifiers in bug prediction: An adaptive method." *IEEE Transactions on Emerging Topics in Computational Intelligence* 1.3 (2017): 202-212.
[56] E. Cambria, et al. "Computational intelligence for natural language processing." *IEEE Comput Intell Mag* 9.1 (2014): 19-63.
[57] A. Chaturvedi and A. Tiwari. "System Network Complexity: Network Evolution Subgraphs of System State series". *IEEE Transactions on Emerging Topics in Computational Intelligence* (2018).
[58] A. Chaturvedi, and A. Tiwari. "SysEvoRecomd: Graph Evolution and Change Learning based System Evolution Recommender". *IEEE International Conference on Data Mining Workshops (ICDMW)* 2018, 1499-1500.

Animesh Chaturvedi is working towards the PhD degree at the IIT Indore. He received the BEng degree from IET - Devi Ahilya 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 with IIT-Kanpur, Motorola, and Arris. He has been reviewer for IEEE TCC, IEEE TETC, IEEE TBD, and IEEE SJ. He has been student volunteer in 36th IEEE/ACM ICSE 2014 and 6th IEEE CloudCom 2014. His research interest is in Data mining, Machine Learning, 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).



Nicolas Spyratos received his BEng degree from the National Technical University of Athens, Greece, his M.Sc. degree from the University of Ottawa, Canada, his Ph.D. degree from Carleton University, Canada and his "thèse d'état" from the University of Paris South 11, France. He worked as a researcher for Bell-Northern Research in Canada, and for INRIA and the National Research Council (CNRS) in France, prior to joining the University of Paris South as a full professor in 1983, where he was heading the database group from 1985 to 2011. He is currently Professor Emeritus at the University of Paris South, scientific advisor of Japan Science and Technology (JST), and affiliated senior scientist at the Institute of Computer Science of Crete, in Greece (http://www.ics.forth.gr/). His research interests include databases, big data analytics, conceptual modelling, and digital libraries. He is the author of over 200 articles in international journals, books and conferences, and has supervised the work of 24 PhD students. He has also served on the program committee of over 100 international conferences, he has participated in over 25 national, European and international research projects and has served as evaluator for the NSF and the European commission.