Hindawi
Complexity
Volume 2017, Article ID 3460919, 24 pages
https://doi.org/10.1155/2017/3460919

Research Article

Automatic Emergence Detection in Complex Systems

Eugene Santos Jr. and Yan Zhao

Thayer School of Engineering, Dartmouth College, Hanover, NH, USA

Correspondence should be addressed to Eugene Santos Jr.; eugene.santos.jr@dartmouth.edu

Received 14 March 2017; Revised 1 July 2017; Accepted 24 July 2017; Published 12 September 2017

Academic Editor: Sergio Gómez

Complex systems consist of multiple interacting subsystems, whose nonlinear interactions can result in unanticipated (emergent) system events. Extant systems analysis approaches fail to detect such emergent properties, since they analyze each subsystem separately and arrive at decisions typically through linear aggregations of individual analysis results. In this paper, we propose a quantitative definition of emergence for complex systems. We also propose a framework to detect emergent properties given observations of its subsystems. This framework, based on a probabilistic graphical model called Bayesian Knowledge Bases (BKBs), learns individual subsystem dynamics from data, probabilistically and structurally fuses said dynamics into a single complex system dynamics, and detects emergent properties. Fusion is the central element of our approach to account for situations when a common variable may have different probabilistic distributions in different subsystems. We evaluate our detection performance against a baseline approach (Bayesian Network ensemble) on synthetic testbeds from UCI datasets. To do so, we also introduce a method to simulate and a metric to measure discrepancies that occur with shared/common variables. Experiments demonstrate that our framework outperforms the baseline. In addition, we demonstrate that this framework has uniform polynomial time complexity across all three learning, fusion, and reasoning procedures.

1. Introduction

Complex systems usually consist of multiple subsystems, whose nonlinear interactions can cause unpredictable and disastrous outcomes. However, it is intractable to analyze all possible outcomes in complex systems directly due to the combinatorial nature of this problem. Extant analysis approaches often build separate models for all subsystems and make conclusions about the entire system by linearly aggregating individual analysis results. This approach, although simple, cannot model emergence of complex systems. In fact, emergence is one of the most challenging concepts of complex systems. However, there exists significant discrepancy about the nature of emergence. Some researchers, such as Mill and John [1] and Broad [2], model complex systems using a layered approach, in which the world consists of different strata. Per this approach, higher-level emergent properties result from lower-level causal interactions. Others, such as Wears et al. [3], study emergent properties by predictive approaches and claim that emergent properties are system level features that could not have been anticipated. Per this definition, emergent properties are those that cannot be predicted even by individuals who possess thorough knowledge of the parts of this system. Popper and Eccles [4] relate emergence to unpredictability by studying the nondeterminism within complex system. Another viewpoint identifies a spectrum of approaches to emergence. Bedau [5] distinguishes between weak and strong emergence. Per his definition, weak emergence can be derived from the knowledge of the system's microdynamics and external conditions but only by simulation. Strong emergence, on the other hand, cannot be derived even by simulation.

No matter how emergence is defined, the consensus among these definitions is that emergence stems from the interaction of subsystems of a complex system. To model subsystem interactions and detect resulting emergence in a complex system, we need to model this complex system first. Extant complex systems modeling techniques can be classified into three groups: (1) subject matter experts (SMEs) manually analyze system dynamics and create a descriptive model, such as the model in [3]; (2) experts simulate system dynamics via agent-based complex systems model (ACS),

such as the model in [6]; and (3) data scientists collect data about individual subsystems, learn subsystem models from data via machine learning approaches, and integrate subsystem models via ensemble methods, such as the model in [7]. The first approach is only useful to perform postevent analysis, since SMEs can only manually analyze event related scenarios from all possible scenarios, whose number is combinatorial. The second approach requires that experts manually build behavioral models for each agent and set up proper parameters. It is both time-consuming and expensive to build such models for large-scale complex systems. The third approach, even though easy to apply and suitable for large complex systems, cannot detect emergence, because ensemble methods integrate subsystem models by their outputs, neglecting their interactions among shared/common variables.

To overcome the drawbacks in extant approaches, we need a new framework that can automatically build complex system models from data, can detect emergence, and can scale to large complex systems simultaneously. The first requirement for the new framework is learning a complex system model from data automatically. Given a single dataset drawn from the entire complex system, we could simply learn a single model in hopes of capturing all interactions and then detect emergence within it. However, since large-scale complex system usually consists of multiple (loosely) coupled (possibly competing) subsystems, it is impractical (and likely infeasible) to construct a single dataset which captures all its features and dynamics. We typically only have access to multiple datasets corresponding to different subsystems at best. Due to this limitation, we can only learn a separate model from each dataset for a subsystem and fuse them into one model via shared variables between different submodels. Ensemble methods learn separate models for different subsystems, but ensemble method infers on these models separately and chooses the (weighted) majority opinion as the final opinion. However, the true result may differ from the majority opinion.

We provide an alternative definition of emergence in complex systems derived as follows: Given some target variable, we query its state on the subsystem models learned from corresponding datasets and group their opinions into majority and minority sets. Then we observe its state at the entire system level. If its true state (observed over the entire system) is different from the majority opinion given by subsystems, we consider this situation as emergent. This is like the one given by predictive approaches in that it "cannot be predicted even by individuals who possess thorough knowledge of the parts of this system."

As such, based on the existence of majority and minority opinions, we can define emergence as composed of four types. If all subsystems form a unanimous opinion, and the true result differs from it, we call it Type 1 emergence. If both majority and minority opinions exist, but the true result differs from both opinions, we call it Type 2 emergence. If both majority and minority opinions exist, and the true result is consistent with the minority opinion, we call it Type 3 emergence. If only minority opinions exist, but the true result differs from all minority opinions, we call it Type 4 emergence. This emergence definition is complete for a complex system with an arbitrary number of subsystems, if each subsystem can provide a valid opinion about the queried target. However, if some subsystem cannot provide direct opinion on target variable but can provide opinion about variables which also exist in other subsystems, its opinion will impact other subsystems' opinions about target variable in an implicit manner. Even worse, if such feedback exists among these subsystems, we will not reach a conclusion easily. Such complex scenario will be studied in the future.

In this paper, we describe our approach to modeling and detecting emergence in complex systems according to our proposed definition of emergence. In brief, we first learn subsystem dynamics through a probabilistic graphical model called Bayesian Knowledge Bases (BKBs) [8] from observations on each subsystem. Then we fuse these BKBs into one BKB via the BKB fusion algorithm [9], which includes interactions among subsystems both probabilistically and structurally sound. We name the fused BKB as FBKB. Lastly, we perform belief updating on the fused BKB (FBKB) to detect emergence in this complex system. The entire framework, which consists of learning, fusing, and reasoning blocks, is named as the *Bayesian Knowledge Fusion for Complex System (BKFCS)*.

Experiments on synthetic datasets show that our proposed method can detect emergence over extant approaches. We also show that our proposed algorithm has polynomial time complexity for all three phases of learning, fusion, and reasoning.

The contribution of this paper is twofold:

(i) It defines four types of emergence in a complex system based on deviations from majority and minority opinions observed from each subsystem, derived from observations/datasets of its subsystems. This quantitative data-driven emergence definition is different from extant descriptive definitions of emergence in that it sets up a concrete boundary between different kinds of emergence. This unique quantitative approach is the first of its kind to the best of our knowledge.

(ii) It designs an automatic emergence detection algorithm based on supervised machine learning techniques. This framework is built upon a probabilistic graphical model named Bayesian Knowledge Base, which not only detects the four types of emergence, but also traces back variable interactions resulting in emergence.

The rest of this paper is organized as follows: We begin with brief backgrounds on Bayesian Knowledge Bases (BKBs), learning BKBs from data, multiple BKB fusion, and belief updating on BKBs. These are the principle components used in our detection framework and algorithm. Next, in Section 3, we formally define emergence in complex systems according to our four proposed types, provide an illustrative example of a complex system, apply our emergence detection framework to this example, explore factors underlying emergence in complex systems with respect to our framework,

and briefly recap the framework and its operation. Having established our framework, we detail our experiments and analyses on synthetically generated complex systems testbeds against extant approaches and our proposed factors and measures.

2. Background

This section first introduces Bayesian Knowledge Bases (BKBs), the building block of our proposed framework. Next, we summarize the BKB learning approach for subsystem data and describe how to fuse such multiple BKBs into one fused BKB, which represents subsystem interaction that can cause emergence. Lastly, we present how to run belief updating on a fused BKB and its role in emergence.

2.1. Bayesian Knowledge Bases (BKBs). Before introducing BKBs, we would like to provide some intuitions behind our choice of building blocks for our framework. Researchers have proposed various methods and modeling strategies to explore different aspects of complex system. In this paper, since the research objective is to detect and explain emergence in complex systems, we opted for probabilistic graphical models, which are powerful tools to explore variable relationships and provide quantitative explanations. In fact, probabilistic graphical models such as Bayesian Networks (BN) [10] and Markov Random Fields (MRF) [11] have been widely applied to model causal relationships and/or interactions among variables in a system. Many researchers also proposed different methods to learn a Bayesian Network or Markov Random Field from data [12–20].

However, neither BNs or MRFs will serve our purpose well. In a MRF, variable connections are undirected, which cannot provide a causal relationship. However, one of our goals is to understand causal relationship in emergence. For BNs, extant methods of fusing multiple BNs into one BN have several drawbacks. First, if two BNs include contradictory information about variable causality direction, extant fusion algorithm requires compromise and consensus regarding this direction [12, 13], which results in unrecoverable information loss. Second, if two BNs contain incompatible variable distributions, a new distribution is created by merging them [12]. Unfortunately, this new distribution no longer represents the observed causal relationships found in the subsystems.

To solve these problems, we apply Bayesian Knowledge Bases [8] into our emergence detection framework. BKBs are an alternative to Bayesian Networks (BNs), by specifying dependence at the instantiation level (versus BNs that are specified only at the random variable level); by allowing for cycles between variables; and by loosening the requirements for specifying complete probability distributions. Figure 1 illustrates a simple BKB.

In general, a BKB K is specified by a set of I-nodes $\mathbf{I}$ (instantiation nodes, rectangles), a set of S-nodes $\mathbf{S}$ (support nodes, circles), and edges $\mathbf{E}$ between $\mathbf{I}$ and $\mathbf{S}$, namely, the tuple $\{\mathbf{I}, \mathbf{S}, \mathbf{E}\}$. In a BKB, a variable is called a component (denoted as C). A BKB does not include an icon for a component; instead it represents all instantiations/states of a component with multiple I-nodes. This is different from

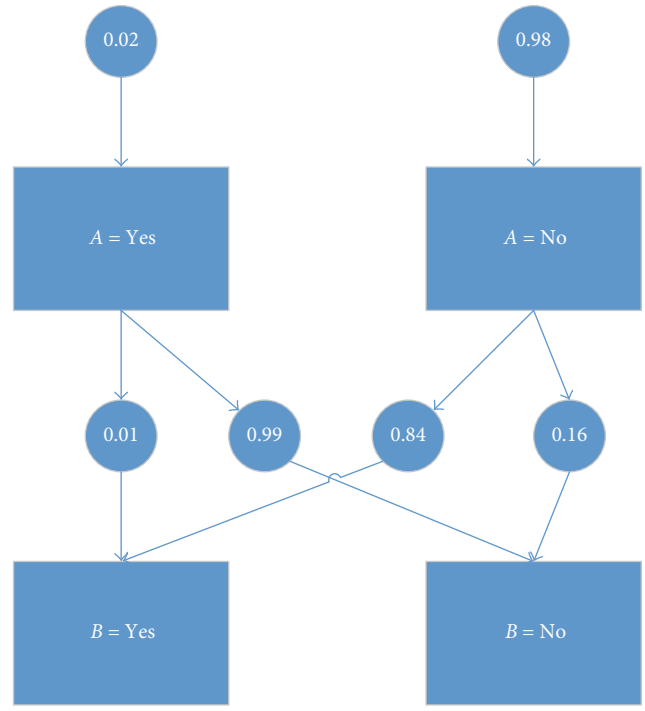

Figure 1: BKB 1.

a BN, which represents a variable/component with a single icon. In Figure 1, there are two components, A and B. Each component can take two states, Yes and No. An I-node I_j^i is noted as a rectangle, and it represents the jth state of the ith variable. In this example, an I-node $I_1^1 : A = \text{Yes}$ corresponds to the first rectangle in the first row with remaining I-nodes $I_2^1 : A = \text{No}$, $I_1^2 : B = \text{Yes}$, and $I_2^2 : B = \text{No}$, respectively.

An S-node is represented as a circle, and it contains a value for some prior or conditional probability. A directed edge connects an S-node and an I-node, which represents direct conditional dependency between the single immediate I-node descendant of the S-node (also called its head, denoted as $\text{head}_K(S)$) and the immediate I-node predecessors (also called its tail, denoted as $\text{tail}_K(S)$). The conditional probability $P(\text{head}_K(S) \mid \text{tail}_K(S))$ is denoted as $\Pr(S)$. In the example, the I-node $I_1^2 : B = \text{Yes}$ is the descendant or head of the S-node with value 0.01, and the I-node $I_1^1 : A = \text{Yes}$ is a predecessor or tail of the same S-node. This connection $I_1^1 \rightarrow 0.01 \rightarrow I_1^2$ represents the conditional probability $P(B = \text{Yes} \mid A = \text{Yes}) = 0.01$. If an S-node only has a descendant but no predecessor, the connection from it to its descendant represents the prior probability. In the example, one such connection is $0.02 \rightarrow I_1^1$, representing that $P(A = \text{Yes}) = 0.02$.

The set of components which set $\{\text{tail}_K(S)\}$ belongs to is the parent component set of I-node $\text{head}_K(S)$, noted as $\text{Pa}(\text{head}_K(S))$. In the example, component A is the parent set of the I-node I_1^2 and the I-node I_2^2. This relationship is similar to that in a BN, where all states of one variable have the same set of parent variables. In a general BKB, however, different states of a component can have different parent variable sets.

This feature allows more flexible variable relationship in a BKB than a BN; however, this is beyond the scope of this paper.

$$\sum_{i\in[1,n]} \left\{ \sum_{j=[1,q(C_i)]} \sum_{s\in\{s|\text{head}_K(s)=I_j^i\}} n\left(I_j^i, \text{tail}_K(S)\right) \cdot \log\left(\Pr(s)\right) - \lambda m \left[q(C_i) \prod_{C\in\{\text{Pa}(C_i)\}} q(C) - \sum_{j=[1,q(C_i)]} \left|\left\{s \mid \text{head}_K(s) = I_j^i\right\}\right| \right] \right\}. \tag{1}$$

2.2. Subsystem Learning from Data. This subsection describes a BKB learning algorithm, inspired by extant BN learning algorithms.

The first step of building our *Bayesian Knowledge Fusion for Complex System (BKFCS)* emergence detection framework is to learn a probabilistic model from subsystem data. In machine learning literature, scoring function-based methods have been widely applied in BN learning problem. Scoring functions can be classified into two categories: information theory-based scoring functions and Bayesian scoring functions [14].

Typical information-theoretic scoring functions include log likelihood (LL), minimal description length (MDL), Bayesian information criterion (BIC) [15], Akaike information criterion (AIC) [16], and mutual information test (MIT) [17]. Typical Bayesian scoring functions include BD [18], BDe [18], BDeu [19], and K2 [20]. However, BKBs represent variable correlations at the variable instantiation level, so we cannot apply existing scoring functions directly to a BKB learning algorithm.

Instead, we propose a modified scoring function designed for learning a BN-like BKB and a greedy algorithm to learn a BKB from a given dataset. This algorithm learns a BKB that maximizes the scoring function (1) given dataset D, assuming it contains m cases and n variables/features, and each feature/component C_i, $i \in [1,n]$, has $q(C_i)$ states/I-nodes. The notation $n(X)$ means the number of cases in which condition "X" holds. The penalty constant λ is set to 0.01 in our algorithm. This function consists of two parts: the first part computes the log likelihood of BKB given dataset D, and the second part is the penalty for complexity and overfitting, which is proportional to the difference between number of possible S-nodes and number of S-nodes that appear in the BKB. The difference between existing scoring functions and our proposed function is that in the penalty term (the second part), MDL, BIC, and AIC only penalize network fitness by total number of parent-child patterns, namely, $q(C_i)\prod_{C\in\{\text{Pa}(C_i)\}} q(C)$.

We have also learned BKBs using Bayes, BDeu, MDL/BIC, and MIT (entropy) and AIC scoring functions and tested their performance against BKBs learned by our proposed function on thirteen UCI datasets. We choose these five popular scoring functions whose usefulness has been widely tested and validated. Their average accuracies are 84%, 83%, 82%, 70%, and 85%, respectively (details in Table 12). Our scoring function achieves 85% average accuracy on the same testbed (details first column in Table 13). It turns out that our function can outperform four of five scoring functions and has comparable performance with AIC. However, BKBs learned using AIC tend to result in simple structures. Even though simple BKBs based on AIC scores can perform equally well in classification tasks compared to BKBs learned based on our proposed method, a BKB learned from AIC score drops variable interactions within a BKB and across BKBs. Without sufficient interactions across BKBs, a BKB learned from AIC score reduces the capability to detect emergence. As such, we cannot use AIC.

In general, learning a BN or BKB from data is NP-hard; therefore we make several tradeoffs to achieve polynomial time complexity. A detailed complexity analysis is provided in Appendix A. In the worst case, the time complexity of the entire learning algorithm is $O(100 * 1000 * n^2m) = O(n^2m)$, where n is the number of variables and m is the number of cases. The other two constants are explained in the Appendix.

To test its performance against other kinds of models on a general supervised classification task on the same testbed, we compare BKB classifier's performance with a wide range of popular classifiers: Adaboost [21], Bayesian Network [22], Sequential Minimal Optimization (SMO) [23], logistic regression [24], and decision tree [25]. Experiment results show that our classifier has comparable accuracy. Since learning a BN-like BKB is not the central contribution of this paper, these results are detailed in Appendix A.

2.3. Subsystem Probabilistic Fusion. This subsection describes how to fuse multiple BKBs learned from multiple subsystem related datasets into one fused BKB (FBKB) that represents the entire system dynamics.

To fuse multiple BKBs, we apply the BKB fusion algorithm developed by Santos Jr. et al. [9]. This resulting fused BKB (FBKB) is the Knowledge Base that the BKFCS framework will reason on.

We design another BKB in Figure 3, which contains the same set of variables, but different probabilistic distributions with BKB 1 in Figure 1. Then we get a fused BKB in Figure 2 by fusing BKB 1 and BKB 2 from Figure 3. Briefly, the idea is to associate each component from each BKB with a special component named as source fusion component. In this example, there are two such components: src$A$ and src$B$, and each has two source I-nodes: "tom" and "john." Each source I-node connects to an I-node via all S-nodes pointing to it. Each source I-node also has one S-node that points to it, representing the reliability/weight of its source. In this example, this weight is 0.5/0.5, meaning that two sources "tom" and "john" are equally reliable.

This source fusion component is the glue that connects variables from different subsystems together. Therefore, it fuses BKBs from various subsystems at the variable instantiation level. In this way, fusion not only computes inferences originated from each subsystem, but also computes new

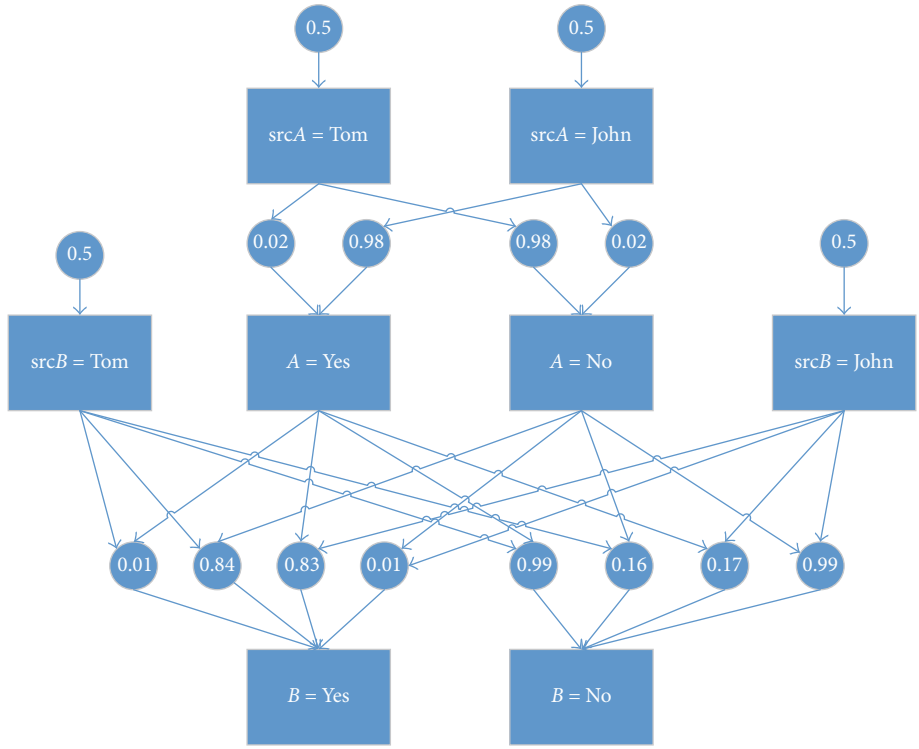

Figure 2: A fused BKB from BKB 1 and BKB 2.

inferences generated by subsystems interactions through their shared variables. The accumulated probability of these new inferences contributes to detection of emergence. Fusion also preserves the distributions and variable relationships in the base subsystems without loss of information. In general, a fused BKB cannot be represented as a BN since both cycles and different parent I-node combinations can occur for each target I-node drawn from the different BKBs being fused together [9]. We provide the details of the BKB fusion algorithm in Appendix B.

The time complexity of BKB fusion algorithm is also polynomial. In particular, its worst-case complexity is $O(|I'|+|S'|+|E'|)$, where $|I'|$ is the number of I-nodes in all subsystem BKBs, $|S'|$ is the number of S-nodes, and $|E'|$ is the total number of edges/arcs. Please refer to Appendix B for details.

2.4. Belief Updating for Emergence Detection. This subsection describes an efficient belief updating algorithm on the FBKB and briefly demonstrates how to detect emergence and perform general classification tasks at the same time.

In general, performing belief updating on a BN or a BKB is an NP-hard problem. It is also NP-hard to find an approximate solution [26]. Bayesian belief updating involves computing the probability that target variable Tar takes a certain state t based on an observation that some other feature variables take certain states. It is denoted as $P(\text{Tar} = t \mid \text{Evidence})$, where Evidence is a set of observed feature variables instantiations. Since it is proportional to the joint probability $P(\text{Tar} = t, \text{Evidence})$, we only compute this joint probability. We can compute this probability by summing up all inferences probabilities which are consistent with Evidence and Tar $= t$. Exact inferencing simply enumerates all inferences, picks out consistent ones, and sums their probabilities as the joint probability.

Table 1: S-node marks.

Source	Mark	Value	Meaning
Tom	s_1	0.02	$P(A = \text{Yes})$
John	s_2	0.98	$P(A = \text{Yes})$
Tom	s_3	0.01	$P(B = \text{Yes} \mid A = \text{Yes})$
Tom	s_4	0.84	$P(B = \text{Yes} \mid A = \text{No})$
John	s_5	0.83	$P(B = \text{Yes} \mid A = \text{Yes})$
John	s_6	0.01	$P(B = \text{Yes} \mid A = \text{No})$

If we do belief updating on BKB 1, BKB 2, and their fused BKB, we will discover emergence. As a simple demonstration of the belief updating procedure, we first name all S-nodes of the three BKBs in Table 1. Notice that, in this example, each pair of S-nodes sums up to 1, so only half of all S-nodes need to be marked. Based on these marks and belief updating rules, we compute variable B's state probability in the three BKBs, as shown in Table 2. In the last two rows, the constant 0.25 is the product of two source fusion variable priors, namely, $0.5 * 0.5$. In fact, since two sources have equal weights, and the constant appears in all inferences, it does not change the relative ordering of B's (two) states' probabilities.

From the last column P, we see that, for both BKBs 1 and 2, $P(B = \text{Yes}) > P(B = \text{No})$. In the fused BKB, on the other hand, we see that $P(B = \text{Yes}) < P(B = \text{No})$. This is one type of emergence, which cannot be detected by aggregating separate analyses on the subsystems. This is just a simple

Table 2: Belief updating of B's state.

B's state	Expression	P
Y_{tom}	$s_1 s_3 + (1 - s_1) s_4$	0.82
N_{tom}	$s_1 (1 - s_3) + (1 - s_1)(1 - s_4)$	0.17
Y_{john}	$s_2 s_5 + (1 - s_2) s_6$	0.81
N_{john}	$s_2 (1 - s_5) + (1 - s_2)(1 - s_6)$	0.18
Y_{fuse}	$0.25 (s_1 + s_2)(s_3 + s_5) + 0.25 (2 - s_1 - s_2)(s_4 + s_6)$	0.42
N_{fuse}	$0.25 (s_1 + s_2)(2 - s_3 - s_5) + 0.25 (2 - s_1 - s_2)(2 - s_4 - s_6)$	0.57

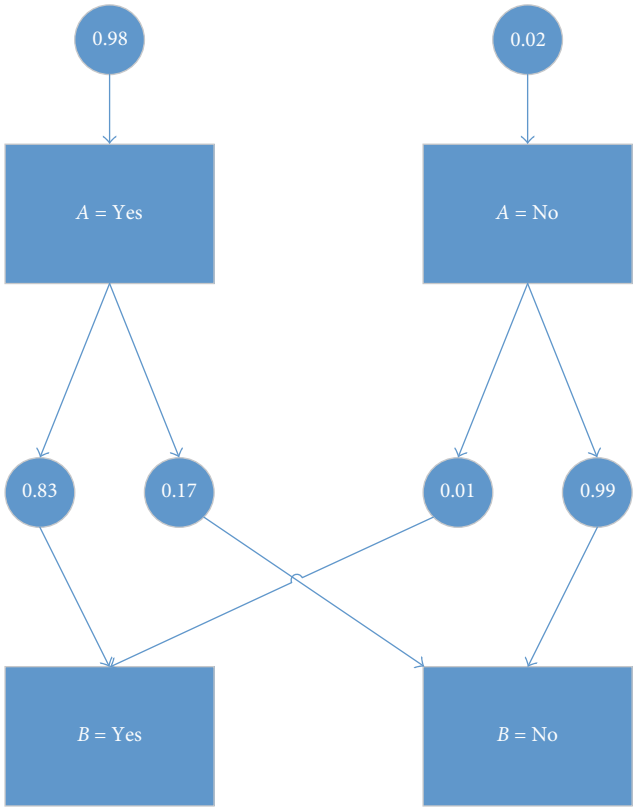

Figure 3: BKB 2.

example of emergence. For general purpose complex systems, we will fully describe our detection framework through real-world examples and provide the underlying mathematical formulations and solutions.

Finally, we note that, in the example fused BKB, the number of inferences doubled compared with that in each single BKB, which is the result of variable interaction. In a fused BKB, there can exist an exponential number of inferences, which makes exact inferencing algorithm extremely demanding with multivariate systems. Instead, we provide a sampling-based approach to approximate the joint probability. To overcome the NP-hard problem, we set up a constant threshold on the number of valid samples we collect before termination. Therefore, our approximation approach has uniform polynomial time complexity and maintains decent performance compared to exact inferencing algorithm. We also compared its running time and accuracy against exact inferencing algorithm and conclude that it is sufficient to serve our purposes for detecting emergence efficiently. In worst case, the time complexity of approximation algorithm is $O(\text{SV} * |\text{Evid}|), r \in N^+$, while the exact inferencing algorithm is $O(a^{\text{SV}} * |\text{Evid}|)$, where SV is the number of shared variables among subsystems, $|\text{Evid}|$ is the number of evidences in a testing case, and a is the average number of states across all shared variables. Details are described in Appendix C.

3. Automatic Detection of Emergence

This section first formally defines different types of emergence in complex systems and explains the intuitions behind these definitions. Next, it applies our proposed framework on a real-world example about a historical US blackout. Lastly, we analyze some major factors causing emergence in a general complex system and how to detect emergence from data automatically. We briefly summarize our proposed emergence detection framework.

3.1. Definition of Emergence in Complex Systems. We define emergence in complex systems formally in this subsection, which forms the basis for all the following subsections.

As mentioned in the Introduction, emergence is unpredictable system behaviors caused by nonlinear interactions within its subsystems. However, many other reasons can cause unexpected/unpredictable system behaviors. In such cases, those unpredictable behaviors should not be categorized as emergence. To rule out alternative explanations of unexpected behavior or emergent behavior of a complex system, such as due to incomplete information, inconsistent measurements, or inexpert judgments, we make three assumptions about this definition:

(i) *Assumption one* is that all subsystems within a complex system are observed, and their features/behaviors are recorded descriptively and/or quantitatively. This assumption indicates that there is no hidden subsystem or obscured subsystem behavior, which may result in unpredictable behavior in the overall system.

(ii) *Assumption two* is that someone with sufficient expert knowledge can build consistent models based on these observables for each subsystem and analyze subsystem behaviors from the constructed models. In this assumption, "consistent" means that the same modeling technique and logic are applied across all subsystems, and no discrimination is allowed.

(iii) *Assumption three* is that we have access to ground truth about both subsystem and overall subsystem

Table 3: Illustration of 4 emergence types.

Sub 1	Sub 2	Sub 3	Whole	Type
a	a	a	b	1
a	a	b	c	2
a	a	b	b	3
a	b	c	d	4

behaviors, so that the emergence definition is based on ground truth, rather than relative metrics influenced due to applied modeling techniques.

In our framework, we require that datasets are available for both subsystems and the overall system and that a maximum likelihood logic is applied in the system behavior modeling. In this way, all three assumptions are satisfied. Prior work [28] studied an emergent border crossing behavior during the 2009 H1N1 pandemic in Mexico using the BKB framework. In that paper, two types of emergence were defined: strong emergence and weak emergence. However, the BKBs were manually constructed from descriptive data sources. In this paper, we apply a data-driven approach for automatic emergence detection whenever data is available.

Given subsystem data and maximum likelihood logic, we can query about target variable's (Tar's) most likely state in all subsystems. Then each subsystem makes decisions based on their partial knowledge of Tar, learned from the corresponding subsystem dataset. The subsystems' opinions can form multiple sets: a majority opinion set and/or minority opinion set(s). In an extreme case, all subsystems form a unanimous opinion, and there is no minority opinion. In another case, each subsystem has a different opinion from the others', or each opinion has an equal number of supporters. In this case, there is no majority opinion.

At last we apply the same logic on an overall system dataset to figure out the most likely state of Tar. Intuitively, if there is a majority opinion from the subsystems side, it is expected to coincide with the overall system opinion. Otherwise, we claim this discrepancy as one form of emergence. If there is no majority opinion from the subsystem side, and overall system opinion agrees with one of the minor opinions, it is also accepted. Otherwise, we also claim it as one type of emergence. Based on these intuitions, we illustrate four types of emergence in Table 3. In this table, Sub 1 to Sub 3 represent three subsystems. Whole means the opinion from overall system. Type labels the type of emergence this case belongs to. The states "a", "b", "c", and so forth represent different opinions about Tar from subsystems and/or overall system.

In general, a complex system can have an arbitrary number of subsystems, but three is the minimum number to have all types of emergence. We notice that not all (if any) will occur in a complex system. If Tar is binary, only Type 1 and Type 3 can occur; if it is multinomial, all four types can occur. Furthermore, per this definition, we believe that Type 3 emergence should be observed most often. The condition for Type 2 emergence is harder to meet, so it should occur less frequently. Type 1 and Type 4 are likely rarest as their conditions are most stringent.

3.2. Emergence Detection: BKFCS. This subsection details emergence detection through BKFCS.

If we have a dataset about system behaviors under various circumstances, we can apply our BKFCS to detect emergence within the system from data. We also name a system configuration as a case in the dataset. A system configuration refers to a variable-state pair tuple, representing system working status. For instance, if a system has two binary variables, X and Y, then it will have at most four different configurations, namely, X = Yes, Y = Yes; X = No, Y = Yes; X = Yes, Y = No; X = No, Y = No. In the system dataset, which is stored as a two-dimensional matrix format, each row corresponds to one configuration, and each column corresponds to a feature/variable in that system. We also call each row an entry or case of the system. In addition, we assume both subsystem datasets and overall system dataset are available. Therefore, we can set up ground truth for each case. To identify an emergent case against a nonemergent case, we need to label each testing case as emergence or nonemergence based on majority and minority opinions. Assuming that subsystem datasets are labeled as D_l, $l \in [1, \#\text{subsystems } n]$, and overall system dataset is labeled as D_c. We use D_c to label ground truth of each case, but only provide BKFCS with subsystem datasets D_l. By comparing its prediction with ground truth, we can measure BKFCS's performance.

To classify a testing case as emergent or nonemergent, we first run belief updating on all BKBs learned from those subsystem datasets. Then we form majority and minority opinions based on individual opinions from all BKBs. Based on these opinions, we know which state of target leads to emergent case and which does not. Next, we perform belief updating again on the fused BKB, which gives probabilities for both emergence and nonemergence states.

$$\begin{aligned} p_+ &= \sum_{t \in \{e | e \text{ is emergent}\}} \Pr(t), \\ p_- &= \sum_{t \in \{f | f \text{ is nonemergent}\}} \Pr(t), \end{aligned} \tag{2}$$

$$\text{th} = p'_+ - p'_- = \frac{p_+ - p_-}{p_+ + p_-}. \tag{3}$$

To simplify this procedure, we first treat emergence detection as a binary classification problem; namely, all types of emergence cases are viewed as positive, while nonemergence cases are viewed as negative. For each testing case, we compute the accumulated probability of this case being positive p_+ and the accumulated probability of it being negative p_- per function (2). Then we normalize p_+ and p_- into p'_+ and p'_- and compare them to determine if this case is emergent per (3). In this equation, if the difference th = $p'_+ - p'_-$ is bigger than a predefined threshold (will be discussed in experiment section), we declare it as emergence. Then we compare claimed result with ground truth to evaluate BKFCS's performance.

3.3. An Example of Emergence in Complex System. This subsection details a real-world emergence example.

We selected the 1996 US west coast blackout [29] as our conceptual demonstration example. On July 2, 1996, a blackout occurred on the west coast of the US, which impacted over two million customers. The first event was a single phase-to-ground fault on the 345 kV Jim Bridger-Kimport line. System protection removed this line from service clearing the fault. Twenty milliseconds later, system protection opened the 345 kV Jim-Bridger-Goshen line due to misoperation of the ground element in a relay at Bridger. Loss of the two lines correctly initiated a remedial action scheme (RAS) that removed two generating units from service. The next event was system protection opening the 230 kV Round Up-LaGrande line due to misoperation of a zone 3 relay at Round Up. These three events together caused a series of disturbances to the entire system and caused overload on other lines, which further brought down more lines offline.

Per incident report [30], "the simultaneous combination of operating conditions on July 2 was not anticipated or studied. The speed of the collapse seen July 2 was not observed in this region and was not anticipated in studies." In fact, due to the combinatorial nature of interactions that could happen in such complex systems, it is impractical to evaluate all combinations in their studies and prevent all possible advert outcomes before they happen.

This incident meets all three assumptions of proposed emergence definition. First, all behaviors and features of each subsystem, which is power supply and delivery system in the case, are recorded. Their designed features are all functional as expected. For each subsystem, its individual purposes, such as line protection, power delivery rebalancing, and overload protection, are all achieved as well. In theory, these measures should be sufficient to protect the entire system from collapsing. In short, this meets the first assumption of no hidden behavior or missing information. Second, all subsystems handle incidents according to the same logic, which is prebuilt into hardware and software action rules. Employees in that company also followed operation procedures to handle all situations they met to solve immediate problems. This satisfies the second assumption of equal treatments in all subsystems. Finally, the entire system behavior is also recorded, which represents system-scale failure. Therefore, we know the ground truth behavior of both subsystems and overall systems.

Since all three assumptions are met, we can claim that the observed behavior belongs to Type 1 emergence. It means that since all subsystems have been reviewed separately, power delivery in the entire network should not fail. However, overall system observation tells us the opposite. In the next subsection, we apply our proposed framework to model this incidence and compute the emergence.

3.4. Applying Emergence Detection Framework on 1996 US West Coast Blackout Incidence. This subsection details how to apply our proposed framework to model this incidence.

In this accident, the first three major events are Jim Bridger-Kimport line open, Jim Bridger-Goshen line open, and Round Up-LaGrande line open. Since details of incidence are recoded in descriptive manner, we manually build three BKBs representing each event (Figure 4). Next, we fuse them into one FBKB (Figure 5) by BKB fusion algorithm. Then we perform BKB belief updating on three-event BKBs and the FBKB and choose variable "system failure" (abbr. "SF") as target. For demonstration purpose, we label S-nodes as before in Table 4. Next, we list target state probabilities for every subsystem (single event) and entire system in Table 5. In the last two rows of this table, variable w_1 is the product of source fusion variable probabilities which correspond to that inference. Remember that the BKBs in this case are simplified such that only instantiated variables states are depicted, so we can see some S-nodes do not occur in any subsystem BKB but occur in overall system BKB.

In Table 5 checking target state probabilities, we know that, in all three events, $P(\mathrm{SF} = Y) < P(\mathrm{SF} = N)$, but in the overall system BKB, we see that $P(\mathrm{SF} = Y) > P(\mathrm{SF} = N)$. This is a Type 1 emergence per our definition. Now we study this emergence from a mathematical point of view. We know the values of S-nodes in these BKBs are just one solution to the following set of inequalities. Other types of emergence can be constructed in a similar way if the feasible region for these inequalities is not empty. This is the mathematical foundation (4) for emergence in this work. However, this real-world example only displays one type of emergence. In the next section, we will discuss emergence detection in a general system.

$$\begin{aligned} &P_d\left(\mathrm{Tar} = \mathrm{Yes}\right) > P_d\left(\mathrm{Tar} = \mathrm{No}\right), \\ &P_{fu}\left(\mathrm{Tar} = \mathrm{Yes}\right) < P_{fu}\left(\mathrm{Tar} = \mathrm{No}\right), \\ &\qquad d = 1 \cdots K \text{ s.t. } \forall i, s_i \in (0, 1). \end{aligned} \tag{4}$$

3.5. Relevant Factors Underlying Emergence. This subsection describes relevant factors effecting emergence in complex systems from data-driven approach and emergence detection on general systems.

Recall the US blackout example above. We noticed that it shared multiple parameters in different subsystems, both variables and probabilities. In a general complex system, however, all kinds of divergence can occur across different subsystems. We now discuss these variations from a data-driven approach, which provides quantitative metrics of these factors.

In some complex systems, different subsystems have similar structures and parameters, such as power delivery system; in other complex systems, subsystems differ from each other, such as in health care delivery systems. It is reasonable to believe that subsystem variation also plays a role in emergence of complex systems. Therefore, if we collect multiple datasets for subsystems of a complex system, we should consider how different datasets coming from different subsystems differ from each other. To quantify their difference, we define dataset similarity metrics. These metrics introduce relevant factors for emergence. These metrics will be used in the Experiments.

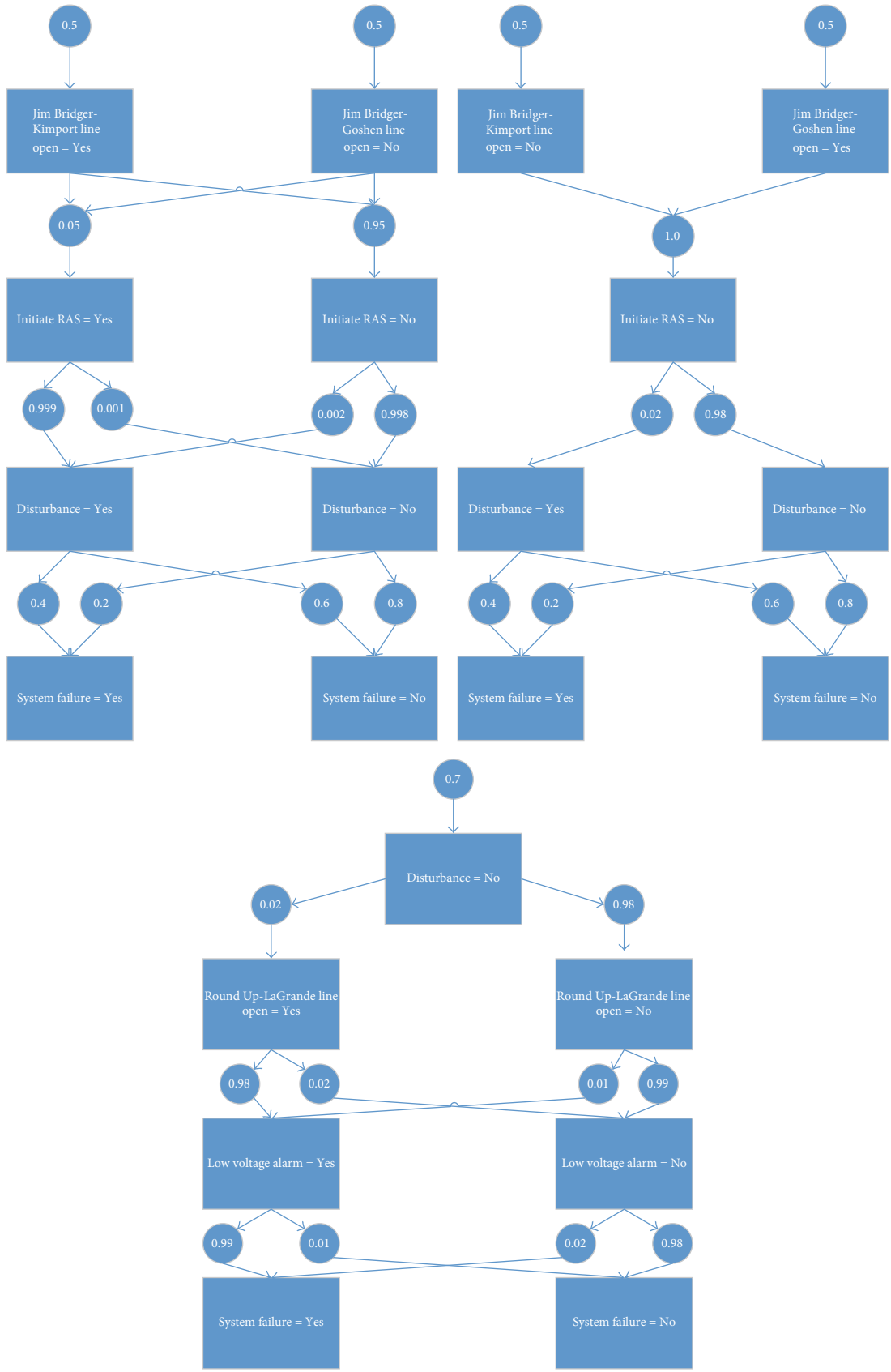

Figure 4: US blackout three-event BKB.

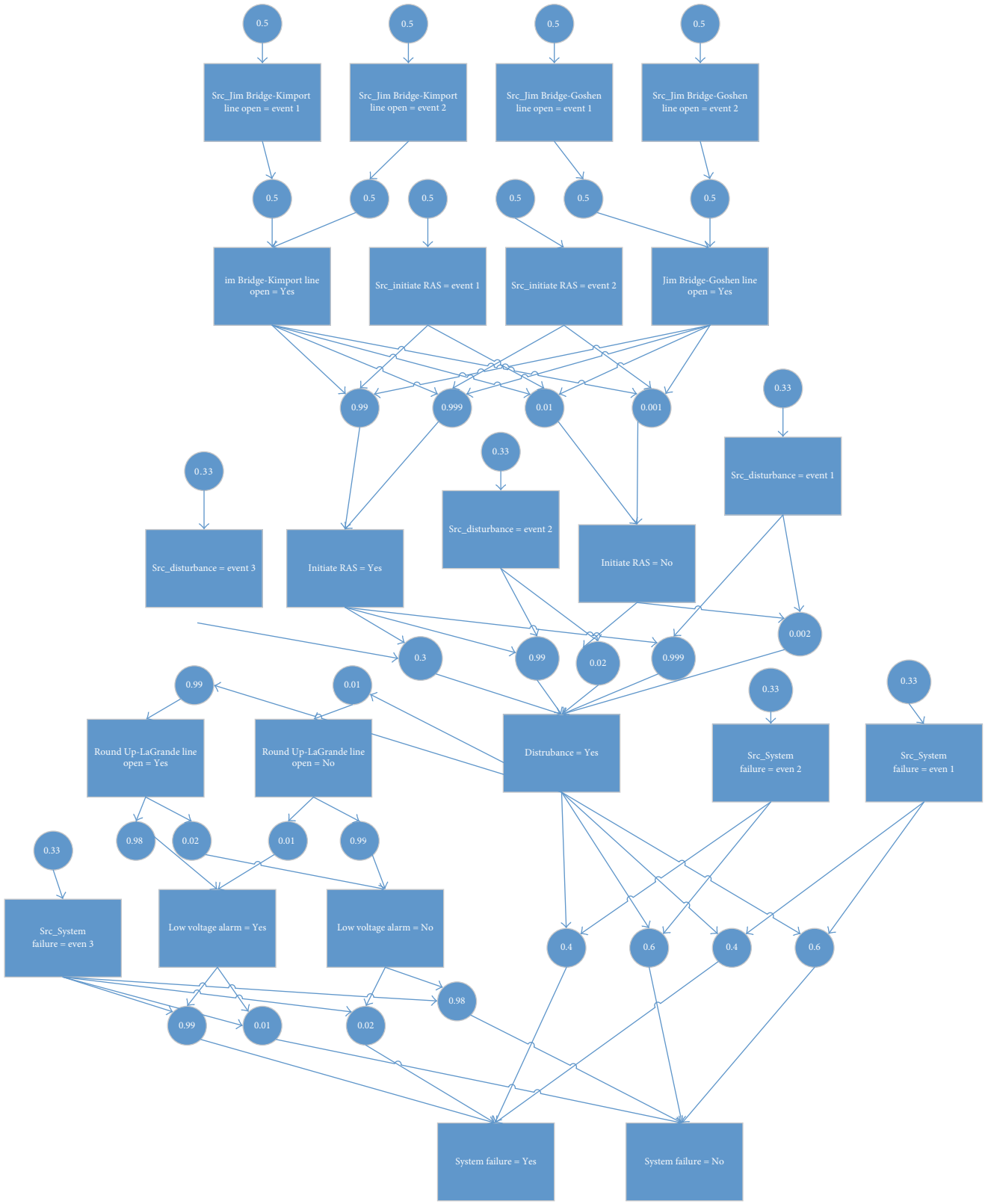

Figure 5: Three-event fused BKB with equal weight.

Table 4: S-node labels for US blackout incidence BKBs.

Event	Mark	Value	Meaning
1, 2	s_1	0.5	Jim Bridger-Kimport line open = Y
1, 2	s_2	0.5	Jim Bridger-Goshen line open = N
1	s_3	0.05	Initiate RAS = Y | (Jim Bridger-Kimport line open = Y, Jim Bridger-Goshen line open = N)
1	s_4	0.999	Disturbance = Y | Initiate RAS = Y
1	s_5	0.002	Disturbance = Y | Initiate RAS = N
1, 2	s_6	0.4	System failure = Y | Disturbance = Y
1, 2	s_7	0.2	System failure = Y | Disturbance = N
2	s_8	1.0	Initiate RAS = N | (Jim Bridger-Kimport line open = N, Jim Bridger-Goshen line open = Y)
2	s_9	0.02	Disturbance = Y | Initiate RAS = N
3	s_{10}	0.7	Disturbance = N
3	s_{11}	0.02	Round Up-LaGrande line open = Y | Disturbance = N
3	s_{12}	0.98	Low voltage alarm = Y | Round Up-LaGrande line open = Y
3	s_{13}	0.01	Low voltage alarm = Y | Round Up-LaGrande line open = N
3	s_{14}	0.99	System failure = Y | low voltage alarm = Y
3	s_{15}	0.02	System failure = Y | low voltage alarm = N
1, 2	s_{16}	0.99	Initiate RAS = Y | (Jim Bridger-Kimport line open = Y, Jim Bridger-Goshen line open = Y)

Table 5: Target state probability in US blackout.

State	Expression	Value
SF = Y in event 1	$s_1 s_2 \left(s_3 \left(s_4 s_6 + \left(1 - s_4\right) s_7\right) + \left(1 - s_3\right) \left(s_5 s_6 + \left(1 - s_5\right) s_7\right)\right)$	0.053
SF = N in event 1	$s_1 s_2 \left(s_3 \left(s_4 \left(1 - s_6\right) + \left(1 - s_4\right) \left(1 - s_7\right)\right) + \left(1 - s_3\right) \left(s_5 \left(1 - s_6\right) + \left(1 - s_5\right) \left(1 - s_7\right)\right)\right)$	0.197
SF = Y in event 2	$\left(1 - s_1\right) \left(1 - s_2\right) s_8 \left(s_9 s_6 + \left(1 - s_9\right) s_7\right)$	0.051
SF = N in event 2	$\left(1 - s_1\right) \left(1 - s_2\right) s_8 \left(s_9 \left(1 - s_6\right) + \left(1 - s_9\right) \left(1 - s_7\right)\right)$	0.199
SF = Y in event 3	$s_{10} s_{11} \left(s_{12} s_{14} + \left(1 - s_{12}\right) s_{15}\right) + s_{10} \left(1 - s_{11}\right) \left(s_{13} s_{14} + \left(1 - s_{13}\right) s_{15}\right)$	0/034
SF = N in event 3	$s_{10} s_{11} \left(s_{12} \left(1 - s_{14}\right) + \left(1 - s_{12}\right) \left(1 - s_{15}\right)\right) + s_{10} \left(1 - s_{11}\right) \left(s_{13} \left(1 - s_{14}\right) + \left(1 - s_{13}\right) \left(1 - s_{15}\right)\right)$	0.667
SF = Y overall	$w_1 s_1 s_2 s_{16} s_4 s_6 + \cdots$	0.109
SF = N overall	$w_1 s_1 s_2 s_{16} s_4 \left(1 - s_6\right) + \cdots$	0.077

Assume V_i and V_j, $i \neq j$, are two sets both including variable v. Let $v^i \in V_i$ and $v^j \in V_j$ be v in V_i and V_j.

Define pairwise variable similarity, $\beta_v(V_i, V_j) = \exp(-(D_{kl}(v^i \parallel v^j) + D_{kl}(v^j \parallel v^i))/2)$, where $D_{kl}(v^i \parallel v^j)$ is the Kullback-Leibler divergence of v^j from v^i.

This measures the difference of a certain variable between two experts' views.

Assume that $v \in I$ and that it only exists in $V_{k_1}, V_{k_2}, \ldots, V_{k_q}$, where $1 < q \leq n$, $1 \leq k_1 < k_2 < k_q \leq n$.

Define variable similarity, γ_v, as the average pairwise variable similarity for variable v in these sets, namely, $\gamma_v = \sum_{1 \leq t < w \leq q} \beta_v(V_{k_t}, V_{k_w})/n(I)$. This measures the difference of certain variable in all experts' views on average.

Define datasets similarity, Ω, as the ratio between $n(I = \{v \mid v \in V_i \cap V_j,\ 1 \leq i \neq j \leq n\})$ and $n(U = \{v \mid v \in V_i \cup V_j,\ 1 \leq i \neq j \leq n\})$, where $n(X)$ denotes number of variables of set X. This measures the difference in the variable selection criteria of two experts.

$$T_l^d = D_l^{j_{\max}}[\text{Tar}], \quad \text{s.t. } j_{\max} = \arg\max_{1 \leq j \leq m} \sum_{1 \leq i \leq n} 1\left(V_{li}^d == D_l^j[V_{li}]\right), \tag{5}$$

$$T_l^{d*} = \arg\max_{t \in \{T_l^d\}} \sum_{j=1}^{m} 1\left(D_l^j[\text{Tar}] == t\right). \tag{6}$$

A related question is how these differences could happen in real-world systems. The answer is complicated. Sometimes different subsystems observe partially overlapped subsets of features on a system, and each shared variable in different subsystems has the same probability distribution. Such systems should have high dataset similarity scores between their subsystems. In other situations, different subsystems observe the same variable from various perspectives, resulting in contradictory probability distributions on each shared variable. These systems will have low dataset similarity scores between their subsystems. In the second kind of situations, shared variables have different distributions from one source to another caused by perspective difference, sample representativeness, random noise, and system biases.

Therefore, once we have datasets about subsystem characteristics under various system configurations, we should be able to identify which configurations lead to emergent behaviors.

As for ground truth, we apply a model independent criterion. Let $V_l = \{V_{l1}, V_{l2}, \ldots V_{ln}, \text{Tar}\}$ be the set of variables

observed in D_l, and let $V_c = \{V_1, V_2, \ldots, V_N, \text{Tar}\}$ be the set of variables observed in D_c. For the dth case $D_c^d \in D_c$, we denote its target state (Tar $= D_c^d[\text{Tar}]$) as Tar^d. Similarly, $\forall V_{li}$, $i \in [1, n]$, we denote its state ($V_{li} = D_c^d[V_{li}]$) as V_{li}^d. Let m be the number of observations in D_l, and let $D_l^j[V_{li}]$, $j \in [1, m]$, $i \in [1, n]$, be the state of variable V_{li} of j^{th} observation in D_l. We determine target state of case D_c^d through function (5), namely, T_l^d. However, if T_l^d is not unique, we pick the best state T_l^{d*} by function (6). Then, for each case D_c^d, $d \in [1, M]$, in D_c, we can compute opinions about target state from all subsystems based on $\{D_l, 1 \le l \le L\}$ and these two equations, resulting in opinion vector $(T_1^{d*}, T_2^{d*}, \ldots, T_L^{d*})$. Combined with its true target state $D_c^d[\text{Tar}]$, we can determine whether it is an emergent case and which type of emergence it belongs to. This forms the ground truth for each case in testing set D_c.

Based on the ground truth, we can perform an emergence detection task. Given several datasets D_l, $l \in [1, L]$, representing subsystem dynamics, we first learn each BKB from one dataset by BKB learning algorithm introduced in the Background. Then we fuse these BKBs into one FBKB per BKB fusion algorithm. Lastly, we run belief updating via sampling method on both individual BKBs and the FBKB for each testing case. To detect emergence versus nonemergence case, we form majority and minority opinions by querying about most probable state of target variable Tar on individual BKBs and compare the opinion of querying FBKB on target variable Tar. Per emergence definition, we decide whether this case is emergence and which type of emergence it belongs to. Finally, we compare our decision with ground truth label to see if we make the right call.

3.6. Emergence Detection Framework Recap. We now provide a step-by-step recap description of our framework.

Step 1. Collect data from multiple subsystems. These datasets contain subsystem feature states as well as target variable states.

Step 2. Learn BKBs for each subsystem via BKB learning algorithm if subsystem data are presented in a structured form. Otherwise, we build BKBs manually based on descriptive data about subsystem features and target variable states.

Step 3. Fuse BKBs for subsystems into one FBKB via BKB fusion algorithm. If we have information about BKB reliabilities, we assign them to fusion algorithm; otherwise, we simply assign equal reliabilities to all subsystem BKBs.

Step 4. Analyze single BKBs and FBKB using belief updating. Compute individual BKB opinions and FBKB opinions for each system feature state combinations.

Step 5. Determine which cases belong to emergence and the emergence type according to definitions in Table 3.

Step 6. Compare BKFCS decision of emergence with ground truth if we have access to it and evaluate its performance.

Table 6: UCI benchmarks characteristics [27].

Dataset	Var	State	Case	Tar
Balance scale	5	23	625	3
Balloon	5	10	76	2
Breast cancer	10	39	286	2
Cars	7	26	1728	4
Hayesroth	5	18	132	3
Monks	7	19	556	2
Nursery	9	32	12960	5
Tictactoe	10	29	959	2
Vote	17	34	435	2
Lymphography	19	63	148	4
Krvskp	35	75	3196	2
Mushroom	23	114	8124	2
Connect 4	43	129	6756	3

4. Experiments

This section begins with designing synthetic datasets that simulate various types of complex systems. Then, it details building complex system models from synthetic dataset via BKB learning and fusion. Finally, we summarize the framework's performance in comparison with existing methods.

4.1. Designing Synthetic Datasets. Even though various types of complex systems exist in real world, the subsystem datasets for emergence modeling typically have not been available for one of two reasons. (1) Extant subsystem behaviors and features are usually described in natural language or equations in postmortem briefings, but we cannot directly apply the framework to such forms of knowledge now. (2) In the cases when subsystem datasets have been recorded, they are not available to the public for commercial, security, or political reasons. As such, we test our proposed framework BKFCS against baselines using synthetic testbeds.

We selected thirteen datasets (Table 6) from UCI machine learning library [27] per several rules. First, both independent and dependent variables are categorical or binary, since BKBs do not currently handle continuous variables. If we choose continuous features and discretize them, we will introduce an uncontrolled level of noise. Second, sample number is sufficient compared to variable number; otherwise, no algorithm will extract useful pattern from that dataset and result in meaningless comparison. Finally, these datasets include various variable and sample number combinations so that they represent a diversity of scenarios—covering different scales of complex systems, different amounts of available data, and various kinds of variable interactions between subsystems and within a subsystem.

To evaluate BKFCS performance, we split one dataset into training and testing set in a 10-fold cross validation fashion. For each training set, we can split it into multiple subsets, where a subset includes a part of all features and all cases. Different subsets have varying numbers of shared/common variables, representing their interactions in complex systems (Algorithm 1). To simulate the dataset similarity difference,

(1) Dataset $\mathbf{D}$ contains variable set $\mathbf{V}$ and target Tar
(2) For Ω in $\{0.1, 0.3, 0.6\}$
(3) # of shared variables: $N_{sv} = \lfloor \Omega * |\mathbf{V}| \rfloor$
(4) For h in $[1, 5]$
(5) Pick N_{sv} variables from $\mathbf{V}$ randomly, assign to $\mathbf{V_s}$
(6) For v in $[1, 10]$
(7) Split $\mathbf{D}$ into $\mathbf{D^v_{tr}}$ and $\mathbf{D^v_{ts}}$, $|\mathbf{D^v_{tr}}| = 90\%|\mathbf{D}|$, $\mathbf{D^v_{ts}} = \mathbf{D} \setminus \mathbf{D^v_{tr}}$
(8) Split $\mathbf{V} \setminus \mathbf{V_s}$ into $\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3}$ evenly
(9) Add $\mathbf{V_s}$ and Tar to $\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3}$
(10) Split D^v_{tr} into $D^{v1}_{tr}, D^{v2}_{tr}, D^{v3}_{tr}$ per $\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3}$
(11) End for
(12) End for
(13) End for

Algorithm 1: Generating original synthetic datasets.

Table 7: Perturbation function to create perturbed sets.

$f_1(x) \sim f_5(x)$	$f_6(x) \sim f_{10}(x)$
$-\log(x)$	x^2
$-\log_{10}(x)$	x^3
e^x	$\sqrt[2]{x}$
$\sqrt[3]{x}$	x^4
$\cosh(x)$	x

we introduce ten popular perturbing functions that transform an original distribution to a perturbed one on shared variables (Table 7).

These functions have various effects on the original distribution: some can transform a uniform or relatively even distribution to a skewed one, others can lessen the skewness of distribution, and others can flip the density of distribution, making rare cases more popular and common ones less popular. In short, they cover most scenarios in which the procedure of fusing multiple inconsistent information can result. The perturbation procedure is as follows: (1) for each shared variable, we compute its original probability mass function (pmf); (2) choose a function randomly for each source; (3) compute the perturbed pmf for each source; and (4) modify shared variable instantiations so that the distribution of the modified shared variable follows perturbed pmf with minimal change.

4.2. Applying BKFCS on One Synthetic Dataset. This subsection demonstrates learning BKBs and BKB fusion from synthetic datasets.

We first demonstrate BKFCS on dataset balloon, where there are 76 cases, and each case includes five variables. Therefore, per algorithm in Algorithm 1, each training set contains 68 cases, and each testing set contains 8 cases. The five variables are "size," "act," "age," "color," and "class" (target variable), all of whom are binary variable. If we set low variable overlap ($\Omega = 0.1$), we can have one shared variable. In one round, we pick "size" as shared feature variable and split the rest three into three subsystems evenly.

Based on three subsets created via this manner, we learn three BKBs via the BKB learning algorithm mentioned in the Background, which are drawn in Figure 6.

Then, we apply the BKB fusion algorithm detailed in the Background to fuse three BKBs into one and perform belief updating on fused BKB. For space reason, we omit showing fused BKB here. Finally, we run emergence detection algorithm on the testing set. The details of emergence detection on it as well as on other datasets will be presented in the following subsection.

4.3. Emergence Detection on All Synthetic Datasets. This subsection details emergence detection algorithm on all synthetic datasets

Here we evaluate BKFCS performance on these datasets. A typical way of evaluating classifier performance is to compare true positive rate against false positive rate and plot the results into ROC figures. To study the ratio of correct claims of emergence versus false claims, we need to know how many cases are truly emergence cases. After all, emergence can only be detected if it occurs in testing sets. We analyze the emergence rate in the synthetic datasets by comparing majority and minority of individual subsystem dataset opinions against overall system opinion on each case in the testing sets. For instance, if, for a test case, three subsets' opinions are the same, but the overall set opinion is different than this opinion, we label this case as Type 1 emergence case. If our model predicts that it has the same opinion of the overall opinion, we classify it as correctly identified; otherwise, we claim it generates a false negative case. To evaluate its overall emergence rate in a dataset, we collapse different types of emergence. The aggregated emergence rate, which sums up all four types of emergence for each dataset under different parameters, is summarized in Table 8.

Perturbation is also involved in some experiments to simulate probability distribution variations in subsystem datasets. We simply named these datasets as perturbed sets and named those which have the same distribution of shared variables as original sets. In most datasets and both original and perturbed sets, emergence rate is positively correlated (with P value < 0.05) to datasets similarity, Ω. This is because

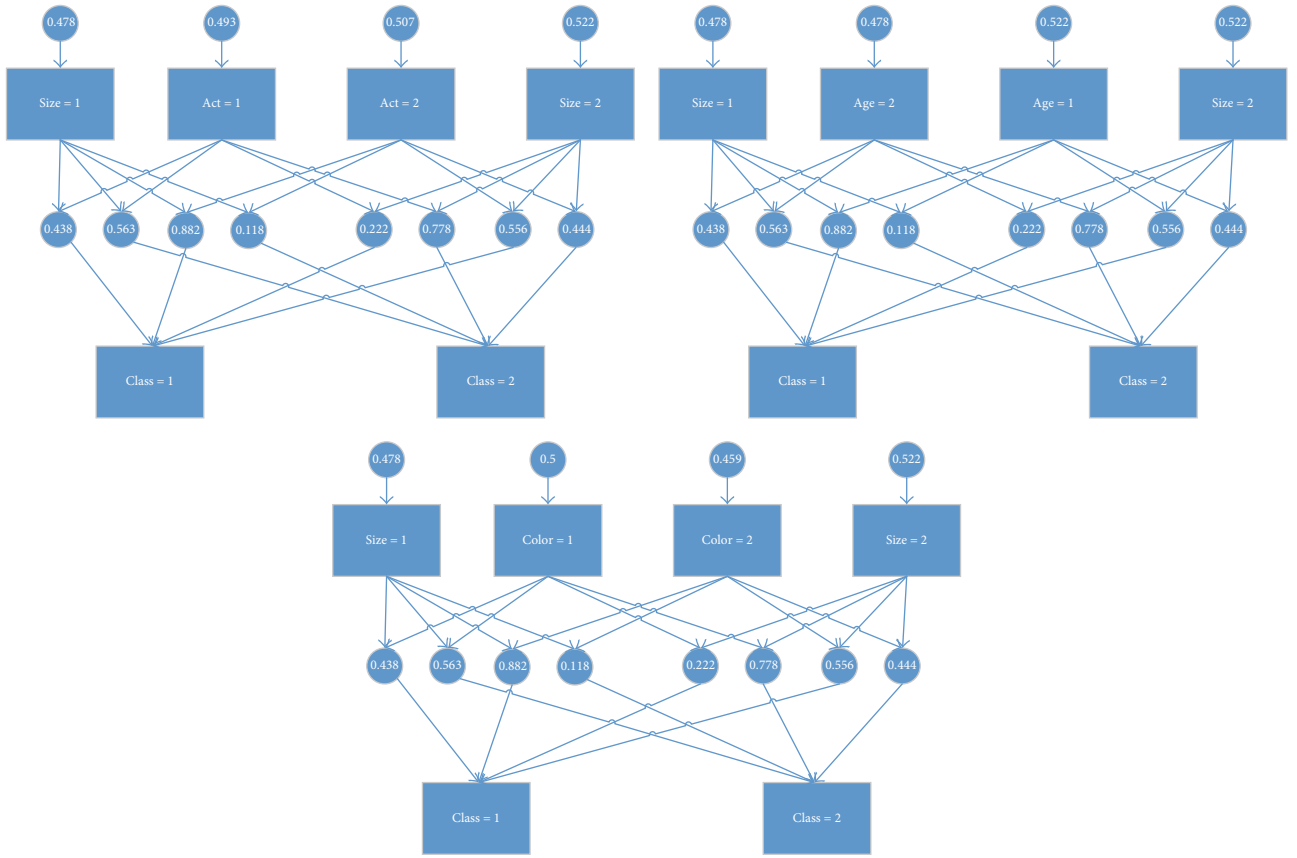

Figure 6: Balloon dataset three-part BKBs.

Table 8: Emergence rate in all datasets.

Set	0.1	0.3	0.6	0.1	0.3	0.6
	Original			Perturbed		
Balance-scale	0.25	0.27	0.28	0.25	0.27	0.31
Balloon	0.34	0.34	0.38	0.34	0.32	0.36
Breast_cancer	0.31	0.43	0.69	0.35	0.60	0.94
Cars	0.28	0.24	0.26	0.28	0.25	0.27
Hayesroth	0.44	0.52	0.62	0.46	0.52	0.56
Monks	0.35	0.29	0.23	0.35	0.3	0.24
Nursery	0.31	0.30	0.25	0.31	0.3	0.25
Tictactoe	0.28	0.28	0.72	0.28	0.31	0.77
Vote	0.20	0.33	0.51	0.21	0.37	0.62
Lymphography	0.44	0.77	0.95	0.57	0.90	0.96
Krvskp	0.20	0.20	0.31	0.51	0.89	0.97
Mushroom	0.01	0.01	0.02	0.12	0.56	0.97
Connect 4	0.35	0.40	0.67	0.73	0.95	0.93
Average	0.29	0.34	0.45	0.37	0.50	0.63

the more shared variables there are among different subsystems, the more interactions exist among various subsystems.

Recall that, in 3.2, we need to compare computed accumulated state difference th in (3) with some predefined decision threshold. In our experiments, we vary this threshold from 0.05 to 0.25 at 0.05 step and list all results. The results for different decision thresholds and different dataset similarities are shown in Figure 7. It only contains results for original sets. We also compute ROCs for perturbed sets and it shows similar relationships, so we omit that due to space limitation. From this figure, we see that, in both original and perturbed sets, all ROCs are above the baseline (this line means "true positive rate" = "false positive rate"). In addition, as Ω grows from 10 percent to 60 percent, most ROC curves move northwest (ensemble method), indicating an improved performance. Thirdly, in most datasets, the decision threshold has a significant impact on precision and recall. Finally, at a fixed threshold, precision and recall have huge variances among different datasets. However, in most conditions, our proposed algorithm can reach 50 percent true positive rate while controlling false positive rate to be under 20 percent.

This figure demonstrates the overall performance of BKFCS on all types of emergence. However, we still want to break it down by each type. Therefore, we need to know emergence rate in each dataset for each type and evaluate its detection efficiency.

Here, we treat the different types of emergence cases separately and show the emergence rate for each dataset in Table 9. In this table, the first column in the first row shows 9%, meaning, in original dataset, when omega is set to 1 (10% overlap features), the average Type 1 emergence rate across thirteen datasets is 9 percent. The second column of

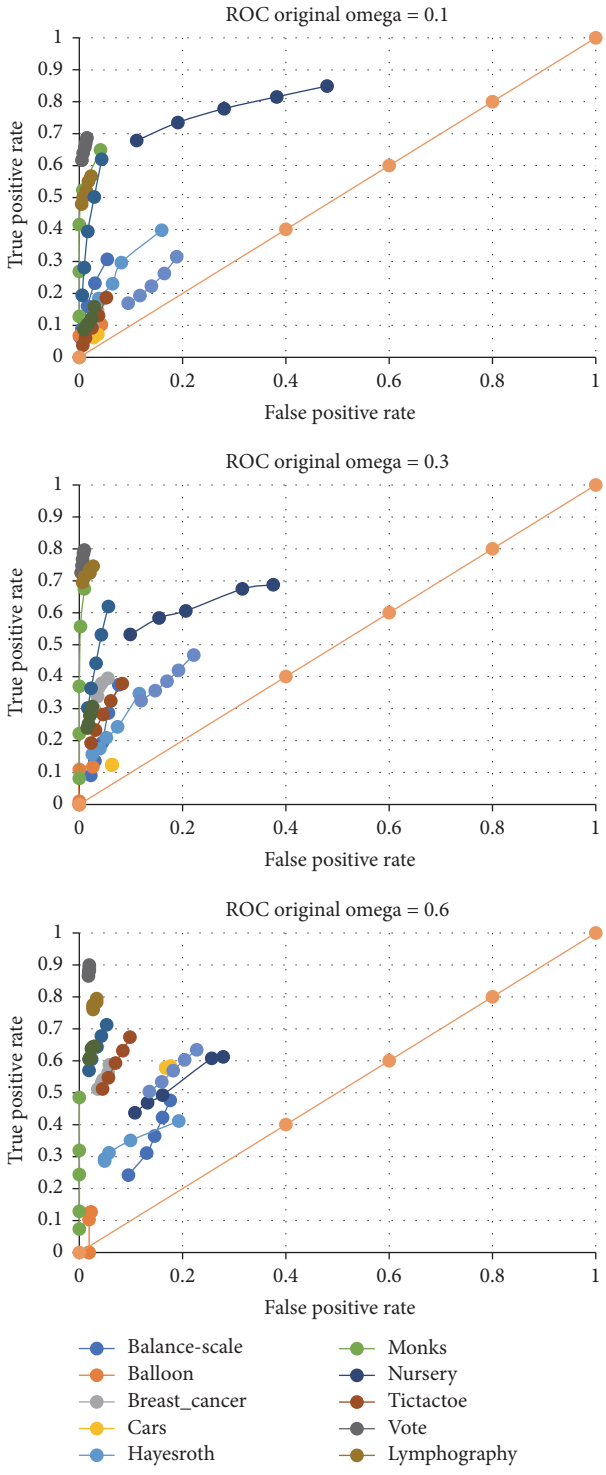

Figure 7: ROC for all parameter combinations.

the first row shows 12%, meaning that the average Type 1 emergence rate for thirteen datasets is 12 percent, and so on. For each omega, Type 3 emergence occurs most often, followed by Type 1 emergence. Type 2 and Type 4 emergence are less often observed. These results are consistent with our emergence definition, because Type 2 and Type 4 emergence indicate more divergent opinions from the various

Table 9: Rate for 4 types of emergence.

Type	Original			Perturbed		
	$\Omega = 1$	$\Omega = 3$	$\Omega = 6$	$\Omega = 1$	$\Omega = 3$	$\Omega = 6$
1	9%	12%	25%	11%	19%	34%
2	6%	11%	11%	9%	19%	19%
3	17%	17%	15%	22%	22%	19%
4	1%	1%	1%	1%	2%	2%

Table 10: Confusion matrix for four emergence types.

Actual type	Predicted type				
	No	1	2	3	4
No	67%	3%	1%	3%	0%
1	2%	10%	0%	0%	0%
2	1%	0%	1%	0%	0%
3	5%	0%	1%	8%	0%
4	0%	0%	0%	0%	0%

subsystems, indicating a harder decision-making process. Type 1 and Type 3 emergence, on the other hand, occur more often in practice, and it should be easier to detect them as well.

To test this hypothesis, we compute a confusion matrix for detection rates on each type of emergence by BKFCS and the average detection rate across thirteen datasets in all omega values in Table 10. In this table, the sum of each row represents the percentage of total cases that really belong to a certain type of emergence. The sum of each column is the percentage of cases that are predicted to be a certain type of emergence. In each column and each row, the number denotes the percentage of cases that is classified as that kind of emergence. The results indicate that BKFCS can detect most Type 3 and Type 1 emergence, but it performs worse on Type 2 emergence. It cannot detect any Type 4 emergence. Its performance is reasonable in that proposed BKB learning algorithm learns a BKB model from subsystem data by maximizing likelihood score, penalized by BKB structure complexity. As a result, it has limited capability in capturing extreme low frequency patterns, which maps to Type 4 emergence.

4.4. Performance Comparison against Ensemble Methods. This subsection compares the performance of BKFCS with BN fusion baselines.

The baseline is set up as follows: for each subsystem dataset, we learn a BN using the Weka machine learning package. Then we learn a BN for the whole system from the union of subsystem dataset. Remember that we only provide classifiers with subsystem dataset and keep whole system dataset as ground truth. Then by comparing majority and datasets with opinion of BN learned from the union dataset, we evaluate its emergence detection capability.

We repeat this procedure on all datasets with all parameters and list true positive rate and false positive rate in Table 11. In comparison, we list BKFCS results in the same table with threshold 0.05 results. At last, we summarize their average performance in six configurations in Figure 8. In

Table 11: BKFCS versus BN performance.

Dataset	Omega = 0.1				Omega = 0.3				Omega = 0.6			
	False positive		True positive		False positive		True positive		False positive		True positive rat	
	BKFCS	BN	BKFCS	BN	BKFCS	BN	BKFCS	BN	BKFCS	BN	BKFCS	BN
Original												
Balance-scale	0.054	0.399	0.306	0.387	0.076	0.026	0.375	0.272	0.176	0.133	0.476	0.067
Balloon	0.043	0.021	0.102	0.227	0.026	0.033	0.118	0.220	0.023	0.043	0.127	0.265
Breast_cancer	0.046	0.471	0.183	0.283	0.055	0.301	0.395	0.221	0.058	0.018	0.588	0.113
Cars	0.036	0.135	0.071	0.235	0.063	0.061	0.124	0.182	0.176	0.076	0.584	0.245
Hayesroth	0.159	0.082	0.398	0.218	0.116	0.042	0.347	0.198	0.193	0.018	0.411	0.218
Monks	0.041	0.055	0.649	0.214	0.010	0.048	0.675	0.194	0.000	0.017	0.485	0.158
Nursery	0.480	0.465	0.849	0.217	0.375	0.463	0.688	0.315	0.279	0.252	0.612	0.200
Tictactoe	0.052	0.441	0.187	0.271	0.083	0.301	0.379	0.206	0.098	0.133	0.674	0.063
Vote	0.015	0.004	0.687	0.009	0.010	0.002	0.797	0.011	0.019	0.004	0.899	0.002
Lymphography	0.023	0.181	0.567	0.289	0.026	0.096	0.746	0.283	0.033	0.000	0.794	0.074
Krvskp	0.043	0.014	0.620	0.122	0.056	0.016	0.620	0.119	0.053	0.015	0.713	0.042
Mushroom	0.029	0.001	0.158	0.171	0.025	0.002	0.307	0.100	0.029	0.004	0.646	0.007
Connect 4	0.189	0.585	0.315	0.461	0.221	0.437	0.468	0.386	0.227	0.257	0.635	0.237
Perturbed												
Balance-scale	0.055	0.403	0.314	0.390	0.073	0.029	0.379	0.279	0.141	0.094	0.533	0.065
Balloon	0.046	0.017	0.122	0.188	0.021	0.042	0.122	0.210	0.030	0.048	0.121	0.185
Breast_cancer	0.052	0.441	0.206	0.303	0.032	0.279	0.448	0.233	0.033	0.057	0.607	0.104
Cars	0.027	0.136	0.057	0.240	0.060	0.059	0.122	0.206	0.155	0.065	0.513	0.236
Hayesroth	0.274	0.082	0.662	0.164	0.221	0.073	0.536	0.135	0.272	0.043	0.549	0.110
Monks	0.036	0.055	0.634	0.202	0.012	0.043	0.686	0.182	0.000	0.015	0.573	0.146
Nursery	0.480	0.465	0.849	0.217	0.377	0.462	0.688	0.314	0.278	0.251	0.613	0.200
Tictactoe	0.059	0.447	0.194	0.280	0.085	0.277	0.431	0.193	0.140	0.099	0.713	0.067
Vote	0.014	0.004	0.679	0.016	0.008	0.002	0.802	0.013	0.029	0.000	0.893	0.007
Lymphography	0.027	0.142	0.563	0.289	0.000	0.022	0.608	0.254	0.020	0.020	0.586	0.121
Krvskp	0.039	0.025	0.703	0.059	0.082	0.045	0.694	0.034	0.120	0.055	0.419	0.034
Mushroom	0.024	0.005	0.674	0.067	0.053	0.007	0.862	0.020	0.273	0.032	0.895	0.006
Connect 4	0.069	0.203	0.498	0.515	0.008	0.023	0.529	0.477	0.031	0.034	0.462	0.283

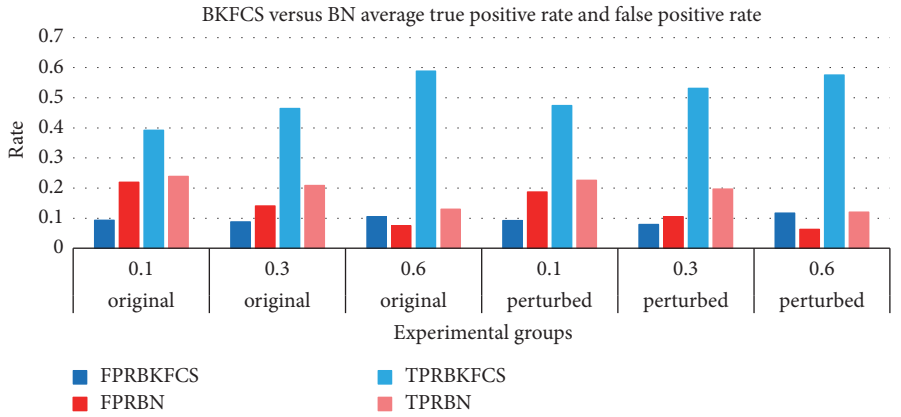

Figure 8: BKFCS versus BN average performance.

this figure, we organize results into six groups from left to right, where group 1 represents original set, omega = 0.1, and group six maps to perturbed set, omega = 0.6. Remember that, for false positive rates, lower is better, and for true positive rate, higher is better. Then we do a one-tail paired t-test for both true positive and false positive rates on all six configurations. Results show that eleven out of twelve tests are significantly different at 0.05 level. All six true positive rates in BKFCS are those of BN, but two are significantly larger than those of BN (groups 3 and 6). Only group 5 shows

no significant difference in false positive rate between two classifiers. In short, BKFCS is much better than BN ensemble approach in detecting emergence in these synthetic datasets results.

5. Conclusion

In this paper, we propose a quantitative definition of emergence and an emergence detection algorithm that learns and fuses several subsystem models through variable interaction, which preserves all inconsistent information. Experiments on synthetic datasets show that this algorithm can better detect emergence in complex systems than extant methods. To the best of our knowledge, this automatic emergence detection approach of fusing graphical models is the first in this field.

Appendix

A. BKB Learning Algorithm

This appendix provides details on a new BKB learning algorithm, analyzes its time complexity, and compares its performance against five baselines on thirteen datasets from UCI machine learning library [27].

We assume that there exists at least one dataset D for each subsystem, from which we learn a BKB. Even though a BKB can contain cycles, we concentrate on learning acyclic BKBs now for simplicity. Learning cyclic BKBs and their modeling impacts will be studied in future work. This algorithm first learns a component level structure G and then builds a BKB K from G and dataset D. In a general BKB, different states of a component i can have different sets of parent components, namely, $\{\text{Pa}(I^i_{j1})\} \neq \{\text{Pa}(I^i_{j2})\}$, $j1 \neq j2$. However, different states of a component have the same set of parent components in a BKB built from G, for G specifies parent-children relationship at variable level other than variable instantiation level. The set of parent components of a component C is denoted as $\text{Pa}(C)$. Based on this simplification, our algorithm learns a BKB that maximizes the score function given dataset D, assuming it contains m cases and n variables/features, and each feature/component C_i, $i \in [1, n]$, has $q(C_i)$ states/I-nodes. The notation $n(X)$ means the number of cases in which condition "X" holds. The penalty constant λ is set to 0.01 in our algorithm. This function consists of two parts: the first part computes the log likelihood of BKB given dataset D, and the second part is the penalty for complexity and overfitting, which is proportional to the difference between maximized number of possible S-nodes and number of S-nodes that appear in the BKB. A nonzero difference means this BKB only associates component C_i with some instantiations of $\text{Pa}(C_i)$ that occur in the training set but cannot generalize to unobserved instantiations, which is a sign of overfitting. What is more, a parent set $\text{Pa}(C_i)$ containing $\prod_{C \in \text{Pa}(C_i)} q(C) > m$ possible instantiations must overfit to the training data, since m is the upper bound of observed patterns.

Based on (1), we design a polynomial time BKB learning algorithm which finds a near-optimal solution, as shown in Algorithm 2. It has been well known that learning a general BN from data is a NP-hard problem, so we set up some constraints to make a polynomial time algorithm possible: first, we include a threshold in the number of iterations (1000); second, we set up an upper bound on the number of parents each feature/variable can have in the BKB (C.1), and it also avoids overfitting per previous analysis about parent pattern limit; last but not least, since this algorithm takes a greedy strategy, it can only find a local maximum from a given starting search point, and we precompute multiple starting points with various density and search for multiple local maxima in parallel. Then we choose the best BKB among these local maxima as an approximation of the global maxima.

The algorithm works as follows: a fully connected DAG (directed acyclic graph) has $n(n-1)/2$ edges/arcs, and we would like to search from multiple initial graphs with different densities. Remember that an "edge/arc" in structure G connects two components/variables, while an edge in BKB K connects an I-node and an S-node. In line (2), we generate a random DAG G_0 with density ratio $* n(n-1)/2$, where ratio ranges from 0.01 to 1 at an interval of 0.01.

To compute it, we first do a random shuffle of variables 1 to n and build a fully connected graph based on this shuffle. Namely, the variable in the front of shuffle points to all variables behind it. It guarantees acyclic property and its time complexity is $O(n)$. Then we pick the first $\lfloor \text{ratio} * n(n-1)/2 \rfloor$ edges from this fully connected graph to form G_0, and its time complexity is $O(1)$. From G_0, we iteratively search for a better graph from all its immediate neighbors. Here an immediate neighbor of G_0 means a graph which can be built by adding, deleting, or reversing an arc/edge (T, H), $H \in [1, n]$, $T \in [1, n]$, $H \neq T$. Here H and T represent two components this edge connects with, while, in a BKB K, $\text{head}_K(S)$ and $\text{tail}_K(S)$ represent I-nodes. There are three possible scenarios, and each scenario corresponds to two potential neighbors, as shown from line (7) to line (11). In each scenario, we test the acyclic property of two potential neighbors through topological ordering in line (13). The time complexity of the acyclicity check is $O(n+ \mid \mathbf{E_{nb}} \mid)$, where $|\mathbf{E_{nb}}|$ is the number of edges in this graph G_{nb}. If a neighbor is acyclic, we compute its score and compare it with the current graph's score $S(G_c)$. In fact, we only compute the scores of nodes in the set $\{V \mid S_V(G_{\text{nb}}) \neq S_V(G_c)\}$ through function (A.3).

In this function, structure graph G is associated with BKB K, and G can be either current graph G_c or its neighbor graph G_{nb}. In line (4), each node score of current best graph G_c is stored as $S_i(G_c)$. In line (14), the change of score Δ is computed as follows: if this neighbor is built by adding or removing an arc (T, H), only H will change its score.

$$\Delta = S_H(G_{\text{nb}}) - S_H(G_c) \tag{A.1}$$

$$\Delta = S_T(G_{\text{nb}}) + S_H(G_{\text{nb}}) - S_T(G_c) - S_H(G_c) \tag{A.2}$$

(1) For ratio in 0.01 : 1 by 0.01
(2) Create a random acyclic graph G_0 with density |ratio $*$ ($n*(n-1)$)/2|, set $G_c = G_0$
(3) For r in 1 : 1000
(4) $S_{\max} = S(G_c) = \sum S_i(G_c)$, $\Delta_{\max} = 0$, $G_{\max} = G_c$
(5) For head in $1 : n-1$
(6) For tail in head $+1 : n$
(7) Edge $E = (T, H)$, Edge $E' = (T, H)$
(8) If $E \in \mathbf{E}(G_c)$, then G_n^1: remove E from G_c, and G_n^2: reverse E
(9) Else if $E' \in \mathbf{E}(G_c)$, then G_n^1: remove E', and G_n^2: reverse E'
(10) Else G_n^1: add E, and G_n^2: add E'
(11) End if
(12) For k in 1 : 2
(13) If $acyclic(G_n^k)$ == true; then
(14) compute $\Delta = \sum_{i \in \{i | S_i(G_{\text{nb}}^k) \neq S_i(G_c)\}} (S_i(G_{\text{nb}}^k) - S_i(G_c))$
(15) If $\Delta > \Delta_{\max}$; then
(16) $\Delta_{\max} = \Delta$, and $G_{\max} = G_n^k$
(17) End if
(18) End if
(19) End for
(20) End for
(21) End for
(22) If $G_{\max}$ == G_c; then
(23) a local maximum $G_{\max}$ is found, $G_r = G_{\max}$
(24) break
(25) End if
(26) End for
(27) If $S(G_r) > S(G_{\text{global}})$; then
(28) $G_{\text{global}} = G_r$
(29) End if
(30) End for

Algorithm 2: BKB learning algorithm.

$$S_V(G) = \left\{ \sum_{j \in V} \sum_{s \in \{s | \text{head}_K(s) = I_j^V\}} n\left(I_j^V, \text{tail}_K\left(I_j^V\right)\right) \cdot \log\left(\Pr(s)\right) - \lambda m \left[q(V) \prod_{C \in \{\text{Pa}(V)\}} q(C) - \sum_{j \in V} \left| \left\{ s \mid \text{head}_K(s) = I_j^V \right\} \right| \right] \right\}. \tag{A.3}$$

Therefore, we compute Δ via (A.1). If this neighbor is built by reversing an arc, both T and H will change their scores, and we compute Δ via (A.2). In both cases, we count all instantiations of $\text{Pa}(C_V)$ in one loop over m cases and compute the log likelihood score for each instantiation. In the worst case, dataset D contains $\min(q(C_i)\prod_{C \in \{\text{Pa}(C_i)\}} q(C), m)$ different patterns. Therefore, the time complexity of computing a node score difference is $O(m + \min(m, q(C_i)\prod_{C \in \{\text{Pa}(C_i)\}} q(C))) = O(m)$. After we get Δ, we compare it with current largest improvement $\Delta_{\max}$ and update its value, as shown from line (15) to line (17). After we evaluate all neighbors of G_c, we update G_c with the best neighbor $G_{\max}$ for the next iteration. However, if no neighbor has a higher score, then G_c is a local maximum, and the iteration stops, as shown from line (22) to line (25). In each iteration, the worst-case time complexity is $O(n^2 m)$. For all iterations from each starting point, the worst-case time complexity is $O(1000 n^2 m) = O(n^2 m)$. The time complexity of entire algorithm is $O(100 * 1000 * n^2 m) = O(n^2 m)$. Therefore, this is a polynomial time complexity algorithm.

In practice, we can optimize running time in several ways: first, we compute a local maximum from different initial graphs in parallel. Second, within each iteration, we compute the node scores of neighbors in parallel. Third, we memorize all node scores for patterns already computed and do a constant-time look up for existing patterns. The platform we use is a 16-node Dell cluster, and each node contains two Intel® Xeon® CPU E5-2640 clocked at 2.6 GHz. Each node has 512 G of RAM. We have a total of 512 hyperthreaded cores/216 physical cores, which can speed up the algorithm by 2 orders of magnitude.

First, we learn BNs from UCI single datasets with five scoring functions using Weka. The classification accuracies are listed in Table 12. In the first column, each abbreviation corresponds to one dataset in the same order as in Table 6. For instance, "Bs" refers to "Balance-scale" and "Co" is short for "Connect 4". The last row "Avg" denotes the average result of all datasets. In the first row, each abbreviation denotes one learning scoring function. Next, we compare

Table 12: Performance comparison of BNs learned by different scoring functions.

	Bayes	BDeu	MDL	MIT	AIC
Bs	91.4	91.4	91.4	74.7	91.4
Ba	71.1	71.1	69.7	76.3	78.9
Bc	74.8	69.6	69.6	69.6	71.0
Ca	93.5	93.8	83.3	20.1	93.6
Ha	67.4	81.1	81.1	67.4	73.5
Mo	74.6	74.6	74.6	91.9	77.5
Nu	94.3	93.3	91.7	0.0	95.3
Ti	80.0	68.2	68.7	76.5	86.2
Vo	95.2	94.9	95.6	92.9	95.4
Ly	81.8	78.4	75.0	72.3	76.4
Kr	97.5	97.1	95.2	96.9	96.8
Mu	100.0	100.0	99.3	99.0	99.7
Co	70.5	67.5	67.2	71.8	69.3
Avg	84.0	83.1	81.7	70.0	85.0

Table 13: Performance comparison on single dataset.

Data	bkbc	ad	bn	smo	lr	dt
Bs	0.90	0.69	0.91	0.90	0.99	0.67
Ba	0.81	0.77	0.77	0.60	0.76	0.65
Br	0.68	0.71	0.73	0.69	0.68	0.73
Ca	0.83	0.70	0.85	0.94	0.94	0.93
Ha	0.78	0.38	0.73	0.80	0.77	0.64
Mo	1.00	0.75	0.75	0.75	0.75	0.96
Nu	0.93	0.66	0.90	0.93	0.92	0.97
Vo	0.94	0.95	0.90	0.96	0.96	0.97
Ly	0.73	0.75	0.85	0.81	0.76	0.80
Kr	0.95	0.94	0.88	0.96	0.98	0.99
Mu	1.00	0.96	0.97	1.00	1.00	1.00
Co	0.65	0.60	0.66	0.74	0.74	0.76
Avg	0.85	0.74	0.82	0.85	0.86	0.84

our algorithm with five baselines. We choose these older algorithms instead of the state-of-the-art ones because the goal is not about learning a classifier that must beat the best performer, but to learn a BKB that can help in detecting emergence. Their 10-CV accuracy is tabulated in Table 13. In the first column, dataset abbreviations are the same as in the previous table. In the first row, each term represents a type of classifier. In particular, "bkbc" means BKB classifier, "ad" means Adaboost classifier, "bn" means Bayesian Network classifier, "smo" means sequential minimal optimization classifier, "lr" means logistic regression classifier, and "dt" means decision tree classifier. The results indicate that our algorithm has competitive performance with other baselines in single source classification tasks.

In addition, we notice that BKBs learned by proposed scoring function also outperform most BNs learned by several extant scoring functions. In fact, according to the "no free lunch theorem" [31], all classifiers have their strength and weakness, as shown in performance variation on various datasets.

B. BKB Fusion Algorithm

This appendix details the BKB fusion algorithm initially designed by Santos Jr. et al. [9] and analyzes its time complexity.

Given L BKBs $\{K_1, K_2, \ldots, K_L\}$ learned from L distinct subsystems, we integrate them into a single BKB K_f that reflects the entire complex system through BKB fusion [9]. Santos Jr. et al. have proven that if all individual BKBs are valid BKBs, the fused BKB is also a valid BKB. This feature means we can build a hierarchy of BKBs representing emergent properties appearing on different levels of complex systems; however, we will not do this in this paper. We first introduce this algorithm and then analyze its complexity.

The BKB fusion algorithm is shown in Algorithm 3. To fuse multiple BKBs, we start with an empty BKB $K_f = \{\mathbf{I}', \mathbf{S}', \mathbf{E}'\}$, and we also need a weighting function representing each BKB's relative importance in a complex system, noted as w in line (1). In line (2), we add all I-nodes from all individual BKBs to $\mathbf{I}'$, add all S-nodes to $\mathbf{S}'$, and add all edges to $\mathbf{E}'$. The time complexity of this combination operation is $O(\sum_{i\in[1,L]} | I_l | + \sum_{i\in[1,L]} | S_l | + \sum_{i\in[1,L]} | E_l |)$. From line (3) to line (11), we add a source fusion I-node for each S-node in all individual BKBs and add a supporting S-node for each source fusion I-node just added. In lines (4) and (5), for each S-node q coming from BKB K_l, we note its head I-node $\text{head}_{K_l}(q)$ as α. In line (6), we add a source I-node i, which connects to I-node α via S-node q. It is an instantiation of source fusion component related to I-node α, noted as S_{R_α}.

We use a source fusion component to distinguish S-nodes from different BKBs that all support the same I-node α. The instantiation $i = (S_{R_\alpha} = \sigma_l)$ means S-node q comes from BKB K_l, and the name of K_l is σ_l. Then we add an S-node q_i pointing to i, which represents the prior probability of this source fusion I-node. Next, we add them into sets I' and S' and add two edges/arcs $q_i \rightarrow i$ and $i \rightarrow q$ to E' in lines (7) and (8). In line (9), we set weight of q in K_f to its weight in K_l. The time complexity of line (3) to line (11) is $O(\sum_{l\in[1,L]} |S_l|)$. In line (12) to line (17), we compute a normalized weight $w'(q_i)$ for each S-node q_i supporting each source fusion I-node i. This weight is also set as the prior probability of this S-node. The complexity of this part is $O(\sum_{S_{R_\alpha}} |\{i \mid i \text{ is a state of } S_{R_\alpha}\}|) = O(L|\{C_f\}|)$, where C_f is a nonsource fusion component of fused BKB K_f. The overall time complexity is $O(|I'| + |S'| + |E'|)$.

C. Sampling-Based BKB Updating Algorithm

This appendix details a uniform polynomial time complexity approximation algorithm for BKB updating, proves its convergence, and compares its performance against an exact inferencing algorithm.

To detect emergent properties in complex systems represented in the form of a fused BKB K_f, we need to run belief updating on it. Bayesian belief updating involves computing the probability that target variable Class takes on state j based on an observation that feature variables take on certain states. It is denoted as $P(\text{Class} = j \mid \text{Evidence})$, where

(1) Let $K_f = \{\mathbf{I}', \mathbf{S}', \mathbf{E}'\}$ be an empty BKB and w' a weight function
(2) $\mathbf{I}' = \bigcup_{l=1}^{L} \mathbf{I}_l,\ \mathbf{S}' = \bigcup_{l=1}^{L} \mathbf{S}_l,\ \mathbf{E}' = \bigcup_{l=1}^{L} \mathbf{E}_l$
(3) For each BKB $K_l,\ l \in [1, L]$
(4) For each S-node $q \in S_l$
(5) Let $\alpha = \text{head}_{K_l}(q)$
(6) Let the source node for q be $i = (S_{R_\alpha} = \sigma_l)$
(7) Add i to I' and add a new S-node q_i to S'
(8) Add the edges $q_i \to i$ and $i \to q$ to $\mathbf{E}'$
(9) Let $w'(q) = w_l(q)$
(10) End for
(11) End for
(12) For all source variables S_{R_α}
(13) Let $\Lambda = \{i | i \text{ is a source node which is a state of } S_{R_\alpha}\}$
(14) Let $\rho = \sum_{i \in \Lambda} r(i)$
(15) for each $i \in \Lambda$, let q_i be the S-node such that $q_i \to i \in E'$
(16) Let $w'(q_i) = r(i)/\rho$
(17) End for
(18) Return $K_f = (K_f, w')$

Algorithm 3: BKB fusion algorithm (Santos et al. 2011).

Evidence is a set of observed feature variable instantiations. Since it is proportional to the joint probability $P(\text{Class} = j, \text{Evidence})$, we only compute this joint probability. We compute this probability by summing up the probabilities of all inferences which are consistent with Evidence and Class $= j$. Exact inferencing simply enumerates all inferences, picks out consistent ones, and sums their probabilities as the joint probability. In general, since BNs are special case of BKBs, exact belief updating is NP-hard [32], and even an approximation of the posterior is NP-hard [33]. However, the emergence detection or general classification task does not require general belief updating. It is possible to design a polynomial time approximation algorithm for our purposes.

$$P_{\text{SI}_s}(\text{Tar} = t) = \prod_{q_i \in \{a | \text{head}_K(a) \in R_s\}} \Pr(q_i) \cdot \prod_{q \in \{b | \text{tail}_K(b) \in E_s \&\& \text{head}_K(b) \in t \cup E_s\}} \Pr(q), \tag{C.1}$$

$$\text{test}(\text{SI}_s) = \left(\sum_{t \in \text{Tar}} P_{\text{SI}_s}(\text{Tar} = t) = 0 ? \text{true} : \text{false}; \right). \tag{C.2}$$

Sampling-based approximation method such as importance sampling, MCMC, and Gibb sampling has been widely applied in BN updating [34, 35]. However, we cannot directly apply an existing approximation method to FBKB. First, a FBKB is not a BN, because it may include cycles and have different parent sets for the same variable. Cycles are introduced when fusing two fragments with conflicting causality graphs: expert one believes that variable V_1 causes V_2 but expert two believes in the opposite. Extant BN approximation methods do not consider these. Second, sampling methods such as importance sampling perform poorly on low frequency samples or extreme CPT entries, while emergence is a rare event which has low frequency. Third, in a fused BKB, a source fusion node is a special kind of node, which represents source reliability beyond an ordinary prior probability. Extant approximation method cannot distinguish them from normal feature variables and therefore cannot fit our special purpose. Therefore, we need to design an approximation method for running belief updating and for emergence detection.

$$\Pr(t) = \sum_{\text{SI} \in \text{SI}_V} P_{\text{SI}}(\text{Tar} = t). \tag{C.3}$$

The approximation algorithm is shown in Algorithm 4. The algorithm works as follows. First, assuming R is the set of source fusion components, we compute the number of combinations of source fusion I-nodes, UB, in lines (2) and (3). Next, we begin to sample $\min(S_{\text{num}}, \text{UB})$ valid inferences from BKB K. In experiments, sample number S_{num} is set to $C * |\text{SV}|$, where C is a constant ranging from 1 to 5. From line (5) to line (9), we create a random vector Q_s for sample s, where Q_s consists of state indices of all components in R, kept in a fixed order. If it has not been visited before, we continue processing it from line (10) to line (19). We instantiate all components in R based on Q_s, denoted as I-nodes set R_s, and all feature components based on evidence set Evid, denoted as I-nodes set E_s, in lines (11) and (12). In line (13), we do belief updating on this sample inference SI_s based on (C.1).

Next, we classify this sample inference as a valid or invalid inference in line (14) based on (C.2) and place its source fusion I-nodes index Q_s into corresponding pools, namely, VP(valid pool) and IP(invalid pool), as shown from line (14) to line (18). After we sample Snum valid inferences or reach the sampling upper bound $\min(\text{UB}, 10 * S_{\text{num}})$, we compute each target state probability by aggregating over inferences related to VP, which is denoted as valid inference set SI_V, through function (C.3) in line (22).

```
(1)  Evidence set Evid, target variable/component Tar, and sample number Snum is given
(2)  Let R = {S_Rα | S_Rα is a source fusion component}
(3)  Let s = 0, UB = Σ_{C∈R} q(C), VP = ϕ, IP = ϕ, P_tm = 0, tm = ϕ
(4)  While (s < Snum)&&(|VP| + |IP| < min(UB, 10 * Snum))
(5)    Index vector Q_s = ϕ
(6)    For each r ∈ R
(7)      Pick a random number j ∈ [1, q(r)]
(8)      Append state index j of component r to Q_s
(9)    End for
(10)   If (Q_s ∉ VP)&&(Q_s ∉ IP)
(11)     Set evidence on feature variables based on Evid, denoted as E_s
(12)     Set evidence on variables in R based on Q_s, denoted as R_s
(13)     Do belief updating for sample inference SI_s
(14)     If test(SI_s) == true
(15)       Add Q_s to VP, s = s + 1
(16)     else
(17)       Add Q_s to IP
(18)     End if
(19)   End if
(20) End
(21) For each state t ∈ Tar
(22)   Compute joint probability Pr(t)
(23)   If Pr(t) > P_tm
(24)     Let P_tm = Pr(t), tm = t
(25)   End if
(26) End for
(27) Output each Pr(t), t ∈ Tar, and report tm as predicted state of Tar
```

Algorithm 4: BKB updating by sampling method.

This sampling upper bound is set up for the following reason: since there are a combinatorial number of inferences in worst case, we must sample a large portion of them before we can get enough valid inferences. This will result in exponential time complexity. We avoid this by setting up a cap on total number of inferences at $10 * S_{\text{num}}$, which guarantees a worst-case polynomial time complexity and still maintains decent relative accuracy according to experiments below. The outputs are the target states with the largest aggregated probability and each state's aggregated probability, in line (27). This sampling approach provides equal sampling frequency for each inference, so that emergence inferences with low probabilities will have a chance of being sampled together with nonemergence inferences with high probabilities, thus overcoming the shortcomings of importance sampling for emergence detection.

Now we consider the correctness of our proposed sampling method. Consider a binary classification problem where all feature variables are observed and where the exact inferencing algorithm predicts $P(\text{class} = \text{Yes}) > P(\text{class} = \text{No})$. An individual inference will provide probabilities for both class states. We call an inference positive if it computes $P(\text{Yes}) > P(\text{No})$, $\Delta = P(\text{Yes}) - P(\text{No})$. Otherwise, we call it negative and the difference is $\Delta' = P(\text{No}) - P(\text{Yes})$. Assume the number of consistent inferences is s and the number of inconsistent ones is t. Since exact inferencing algorithm predicts that $P(\text{Yes})$ wins, it follows that $P(\text{Yes}) - P(\text{No}) = \sum_{i=1}^{s} \Delta(i) - \sum_{j=1}^{t} \Delta'(j) = s\overline{\Delta} - t\overline{\Delta'} > 0$, where $\overline{\Delta}$ is the mean of positive inference prediction value differences.

Based on these definitions, we consider the task where we have sv source fusion components. Each has three source fusion I-nodes. Exact inferencing algorithms need to compute 3^{sv} inferences in the worst case. The percentage of positive inferences is $p = s/(s + t)$. If we sample a sufficient amount $S \ll 3^{sv}$ of valid inferences, then the number of positive inferences follows a binomial distribution $\sim B(p, S)$. The number of negative inferences follows $\sim B(1 - p, S)$. The expected probability mass will be $Sp\overline{\Delta}$ for positive inferences and be $S(1 - p)\overline{\Delta'}$ for negative ones. If we replace p with $s/(s + t)$, we get a positive probability mass $S(s/(s + t))\overline{\Delta}$ and a negative probability mass $S(t/(s + t))\overline{\Delta'}$. Since we already know $s\overline{\Delta} - t\overline{\Delta'} > 0$, we conclude that sampling can give the same prediction as an exact algorithm with a high probability that is positively correlated to size S.

However, the mean of sampled inferences only converges to the real mean $\overline{\Delta}$ or $\overline{\Delta'}$ if sufficient samples are collected, which is caused by the variation in the stochastic process, but it is independent of the valid inference distribution. If we sample inferences with replacement, the sample mean follows a normal distribution $N(\mu, \sigma^2/S)$, where μ and σ are mean and standard deviation of the original inference distribution. If we sample inferences without replacement, the sample mean still follows a normal distribution $N(\mu, \sigma^2((N-S)/(N-1)))$, where N is the total number of valid inferences, and

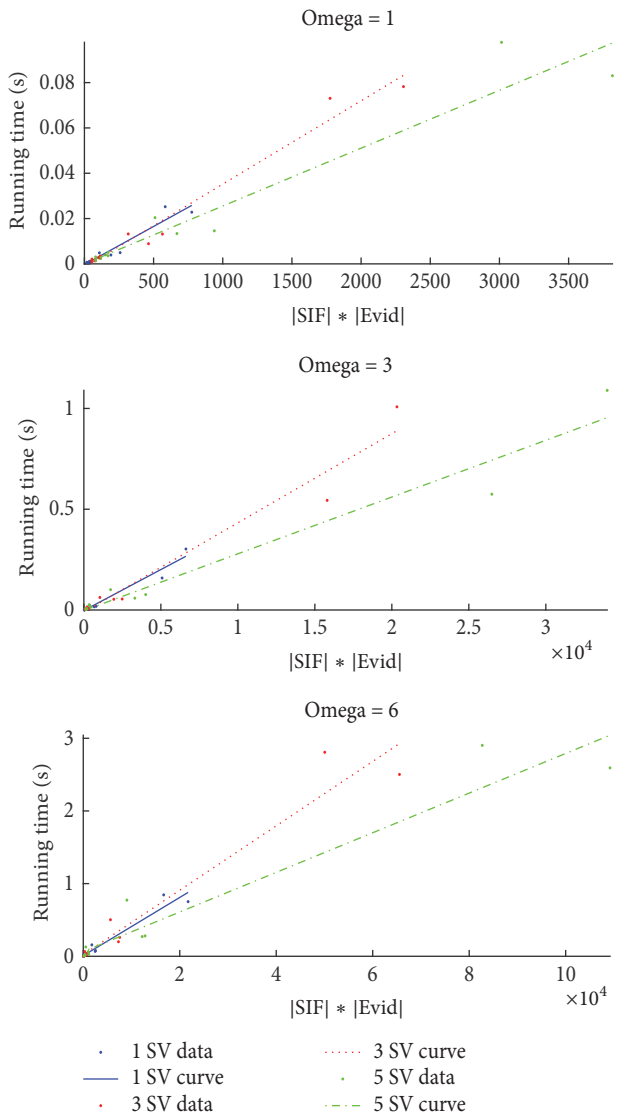

Figure 9: Running time comparison of approximation and exact inferencing.

the other parameters are the same as before. In practice, this approximates a normal distribution pretty well when sample size reaches 30, which is a theoretical number independent of sample size given that the actual population size is much larger than that. This number works well for large BKBs with hundreds or more inferences. In short, this number is much smaller than actual inference size on large BKB. On a small BKB, however, 30 samples are too many relative to the valid inference size. Therefore, we propose a linear sample size $C * |\text{SV}|$, where C is a constant ranging from 1 to 5. We also set up an upper bound on total sampled inference at $10 * C * |\text{SV}|$. In addition, we know that inferencing time for one inference is a linear function of the number of evidences in this inference (number of I-nodes set as evidence); therefore, the time complexity of approximation algorithm is $O(\text{SV} * |\text{Evid}|), r \in N^+$, while the exact inferencing algorithm is $O(a^{\text{SV}} * |\text{Evid}|)$, where a is the average number of states for source fusion nodes. To evaluate our algorithm's performance with various sampling sizes, we test its running time on original sets against an exact inferencing algorithm under various conditions, as shown in Figure 9. We also computed this on perturbed datasets with similar results.

We run our approximation algorithm with 5 different sampling rates $C \in [1, 5]$. In this figure, x-axis is the product of sampled inference number (SIF) and the average number of evidences in one inference, which is $|\text{SIF}| * |\text{Evid}|$; y-axis is the average running time in second for each case/observation. We apply a linear model to fit these two values. The average goodness of fit measure (adjusted R^2) is 0.94 for different C values in original set and 0.96 for perturbed set. Two algorithms have comparable running times on small datasets. However, as the variable number becomes large, the advantage of our sampling approach becomes obvious. The advantage also becomes more obvious when the omega value increases. Therefore, it

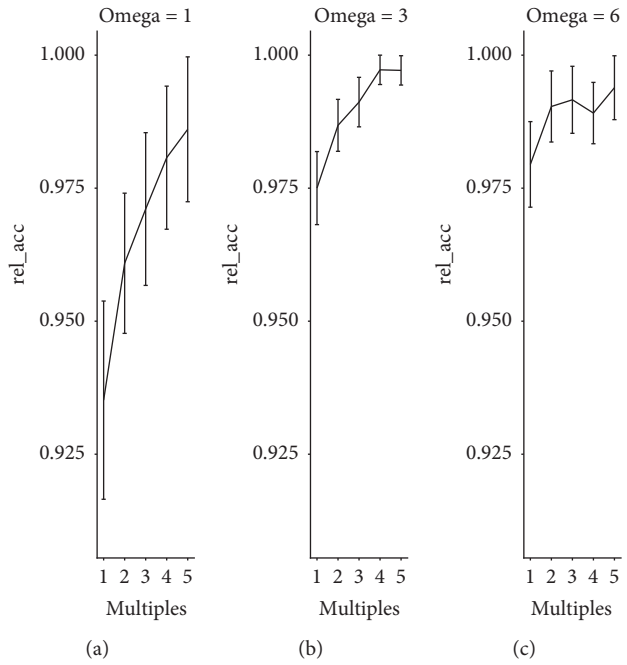

Figure 10: Relative accuracy.

is helpful to apply our sampling algorithm when the fused BKB has many variables, and/or it has many overlapping variables.

We compute the average relative accuracy over the thirteen datasets with three omega levels. The results are shown in Figure 10. In general, the relative accuracy increases as we increase sample size, but even if we sample only twice the number of shared variables, we can already reach more than 95% relative accuracy. Therefore, for large BKBs with thousands of inferences, we recommend a sample size of twice the number of shared variable as the best choice. For small BKBs with less than one hundred inferences, it is easier to compute all inferences directly through brute force. Besides, the standard error is less than 2% across all sample rates and omega values, which indicates the stability of approximation algorithm on various distributions of inferences.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This project was supported in part by AFOSR Grant no. FA9550-15-1-0383, ONR/Naval Postgraduate School Research Initiative Grant no. N00244-15-1-0046, and a DoD STTR with Securboration, Inc.

References

[1] S. Mill and John, *A System of Logic: Ratiocinative and Inductive*, 1843.

[2] C. D. Broad, *Scientific Thought*, Routledge & Kegan Paul, London, UK, 1923.

[3] R. L. Wears, R. I. Cook, and S. J. Perry, "Automation, interaction, complexity, and failure: a case study," *Reliability Engineering and System Safety*, vol. 91, no. 12, pp. 1494–1501, 2006.

[4] K. R. Popper and J. C. Eccles, *The Self and Its Brain*, Springer, Berlin, Germany, 1977.

[5] M. A. Bedau, "Weak Emergence," *Noûs*, vol. 11, pp. 375–399, 1997.

[6] V. Grimm, E. Revilla, U. Berger et al., "Pattern-oriented modeling of agent-based complex systems: Lessons from ecology," *Science*, vol. 310, no. 5750, pp. 987–991, 2005.

[7] K.-C. Ng and B. Abramson, "Probabilistic multi-knowledge-base systems," *Applied Intelligence*, vol. 4, no. 2, pp. 219–236, 1994.

[8] E. Santos Jr. and E. S. Santos, "A Framework for Building Knowledge-Base Under Uncertainty," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 11, no. 2, pp. 265–286, 1999.

[9] E. Santos Jr., J. T. Wilkinson, and E. E. Santos, "Fusing multiple Bayesian knowledge sources," *International Journal of Approximate Reasoning*, vol. 52, no. 7, pp. 935–947, 2011.

[10] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Elsevier, 1988.

[11] R. Kindermann and J. L. Laurie, "Markov random fields and their applications," *Contemporary Mathematics*, vol. 1, Article ID 211739, p. 142, 1980.

[12] I. Matzkevich and B. Abramson, "The topological fusion of bayes nets," in *Proceedings of the Eighth international conference on Uncertainty in artificial intelligence*, pp. 191–198, 1992.

[13] Y. Zhang, K. Yue, M. Yue, and W. Liu, "An Approach for Fusing Bayesian Networks," *J. Inf. Comput. Sci*, vol. 8, no. 2, pp. 194–201, 2011.

[14] A. Carvalho, "Scoring functions for learning bayesian networks," *Inesc-id Tec. Rep*, pp. 1–48, 2009.

[15] W. Lam and F. Bacchus, "Learning Bayesian belief networks: an approach based on the MDL principle," *Computational Intelligence*, vol. 10, no. 3, pp. 269–293, 1994.

[16] H. Bozdogan, "Model selection and Akaike's information criterion (AIC): the general theory and its analytical extensions," *Psychometrika. A Journal of Quantitative Psychology*, vol. 52, no. 3, pp. 345–370, 1987.

[17] L. M. de Campos, "A scoring function for learning Bayesian networks based on mutual information and conditional independence tests," *Journal of Machine Learning Research (JMLR)*, vol. 7, pp. 2149–2187, 2006.

[18] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data," *Machine Learning*, vol. 20, no. 3, pp. 197–243, 1995.

[19] W. Buntine, "Theory refinement on Bayesian networks," in *Proceedings of the Seventh Conf. Uncertain. Artif*, pp. 52–60, 1991.

[20] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, no. 4, pp. 309–347, 1992.

[21] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, part 2, pp. 119–139, 1997.

[22] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[23] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Adv. Kernel Methods*, vol. 12, pp. 185–208, 1999.

[24] S. L. Gortmaker, D. W. Hosmer, and S. Lemeshow, "Applied Logistic Regression," *Contemporary Sociology*, vol. 23, no. 1, p. 159, 1994.

[25] P. H. Swain and H. Hauska, "Decision tree classifier: design and potential," *IEEE Trans Geosci Electron*, vol. 15, no. 3, pp. 142–147, 1977.

[26] T. Rosen, S. E. Shimony, and J. Santos, "Reasoning with BKBs—algorithms and complexity," *Annals of Mathematics and Artificial Intelligence*, vol. 40, no. 3-4, pp. 403–425, 2004.

[27] Bache and M. Lichman, "UCI Machine Learning Repository," vol. 2008, no. 14/8, 2013.

[28] E. E. Santos, E. Santos, J. Korah et al., "Modeling emergent border-crossing behaviors during pandemics," in *Proceedings of the SPIE Defense, Security, and Sensing*, vol. 8711, p. 87110Z, Baltimore, Maryland, USA.

[29] L. Pereira, "Introduction and background to Synchronous Unit Testing and Model Validation in the WSCC," in *Proceedings of the IEEE Power Engineering Society. 1999 Winter Meeting (Cat. No.99CH36233)*, pp. 151–156 vol.1, New York, NY, USA, Feburary 1999.

[30] WECC, Western interconnection (WSCC) system disturbances, 1996.

[31] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[32] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence. An International Journal*, vol. 42, no. 2-3, pp. 393–405, 1990.

[33] P. Dagum and M. Luby, "Approximating probabilistic inference in Bayesian belief networks is NP-hard," *Artificial Intelligence*, vol. 60, no. 1, pp. 141–153, 1993.

[34] H. Guo and W. Hsu, "A survey of algorithms for real-time Bayesian network inference," *Jt. Work. Real-Time Decis. Support Diagnosis*, 2002.

[35] J. L. Beck and S.-K. Au, "Bayesian updating of structural models and reliability using Markov chain Monte Carlo simulation," *ASCE Journal of Engineering Mechanics*, vol. 128, no. 4, pp. 380–391, 2002.