Reasoning about the disruption patterns for train system using Bayesian Network and Prolog

Yunita Rachma Pradiawati¹, Yanti Rusmawati² and Muhammad Arzaki³
Computing Laboratory, School of Computing, Telkom University, Indonesia
E-mail: ¹yunitarp@outlook.com, ²rusmaway@telkomuniversity.ac.id, ³arzaki@telkomuniversity.ac.id

Abstract. We construct a Prolog-based expert system to reason about the disruption patterns for train system using Bayesian network and Prolog. The disruptions dependencies are modeled using Bayesian network and the reasoning is carried on using Prolog. We choose Bayesian network because it is one of the most efficient and elegant framework to represent and reason probabilistic graphical model. The causative relationship among disruptions is represented using Directed Acyclic Graph (DAG). We use Prolog to improve the efficiency of the reasoning process by defining Bayesian network and its probabilistic information into a knowledge base. The causative relationships among disruptions are also modeled in terms of Prolog rules. Our Prolog-based expert system combines the statistical reasoning capability of Bayesian network and logic programming efficiency. The system provides comprehensive reasoning regarding the causative probability of events, the causative relationship among disruptions, as well as the most triggering and triggered disruptions in train system.

1. Introduction
Electric train or also known as Commuter Line is one of the most important public transports in Jakarta and its surrounding area. According to [1], until August 2017, the average daily Commuter Line passengers reached 993,804 people, with the record of the highest daily passengers is 1,066,522. One of the prominent reasons why the majority of daily travelers prefer to use Commuter Line rather than other public transportations is its relatively economical ticket price. In addition, the passengers can also reach to destination place on-time without worrying about traffic congestion. Nevertheless, it still has many drawbacks, such as the frequent disruptions that happen almost every day, e.g.: overhead line and pantograph disruptions. PT. Kereta Commuter Indonesia—as the principal operator of the train system—recorded 422 disruptions which occurred in 2015 [2]. These disruptions can originate from inside or outside Commuter Line system. When a disruption happens, a Commuter Line’s officer needs to trace the causes of the disruption and check whether it may lead to other disruptions. Consequently, there is a logical path from a disruption to other disruptions which also represents the causative relationship among disruptions. Here, we refer to the collection of these logical paths as disruption patterns. Accordingly—from formal data analysis perspective—it is necessary to build a model which is capable in representing the causative relationship among disruptions in train system.
In 2016, Rusmawati and Rismala [3] has developed a model of the disruption patterns of train system in Indonesia using Bayesian network. The authors showed that a disruption may lead to or caused by other disruptions. Unfortunately, their disruption patterns do not consider the part-whole relation of the train system, which is essential for properly ensuring that the model represents the cause and effect of disruptions in the entire train system.

Bayesian networks are one of the most efficient and elegant frameworks for representing and reasoning probabilistic models [4]. They have been applied to many real-world problems in diagnosis and forecasting [5]. One can observe cause and effect relationship clearly through a Directed Acyclic Graph (DAG) in Bayesian network. Moreover, they can handle uncertainty by means of probability theory. However, Bayesian networks have limitation regarding the implementation of their inferences [6]. The complexity of the inferences in Bayesian networks limits the size of models that can be effectively reasoned over [6]. Several research [7, 8, 9, 10, 11] address this problem by integrating Bayesian networks and logic programming by including probabilistic information into a knowledge base.

Prolog is one of the most powerful and flexible logic programming frameworks [12, 13, 14]. In Prolog, one can easily express the relationship among objects as well as develop rules to reason this relationship. This feature makes Prolog a powerful language for Artificial Intelligence problem-solving and non-numerical programming in general. There are well-known examples of symbolic computation whose implementation in other standard imperative languages took a lengthy incomprehensible code. However, when this algorithm implemented in Prolog, the algorithm becomes more compact and easily understood [12].

In this research, we construct a Prolog-based expert system to reason about disruption patterns for train system using Bayesian network and Prolog. Bayesian network is used to reason about the probability when a disruption happens, while Prolog is utilized to reason about the causative relationships among the disruptions. We also add part-whole relation facts regarding the train system to reason about the influence of a disruption to the entire train system. As a forthcoming benefit, PT. Kereta Commuter Indonesia can use our expert system to calculate the probability of a disruption and trace the cause and effect when a disruption happens.

The rest of this paper is organized as follows. In Section 2 we briefly review the methods to represent Bayesian networks in Prolog. In Section 3 we present the technique and process to construct Bayesian networks of disruption patterns in train system. We provide Prolog codes for the previously constructed Bayesian networks as well as other rules pertinent to their reasoning in Section 4. In Section 5 we describe some queries implementation of Prolog rules in our Prolog-based expert system. Finally, the conclusion and future works are discussed in Section 6.

Availability Our Prolog-based expert system as well as its manual and pertinent documentations are available at [https://github.com/yunitarp/ReasoningTrainDisruptionWithProlog/](https://github.com/yunitarp/ReasoningTrainDisruptionWithProlog/).

2. Representation of Bayesian Network in Prolog

We first briefly discuss the representation of Bayesian Network in Prolog. We assume that the readers already have basic knowledge of probability theory, Bayesian network, and logic programming in Prolog. We refer the readers to [12 Section 15.6] for more comprehensive discussion regarding the representation of Bayesian network in Prolog. The explanation of the representation of Bayesian network in Prolog has been comprehensively discussed in there. We adopt those representations to model disruption patterns for train system. For example, suppose we have a simple Bayesian network of five random variables as in figure 1. This network illustrate the cause and effect when fallen tree and fire disruptions happen and they cause overhead line, pantograph, and static inverter disruptions. We denote the disruptions fallen tree, fire, overhead line, pantograph, and static inverter as Ft, Fi, OL, P, and Si, respectively.
Figure 1. A simple Bayesian network of five random variables. When a fallen tree disruption occurs, it likely triggers an overhead line disruption, which in turn prompts pantograph and static inverter disruptions. The fire disruption also triggers the overhead line disruption.

The Conditional Probability Table (CPT) of the Bayesian network in figure 1 is described in table 1. The presented probability is obtained by calculating the disruptions data using Parameter Estimation in Bayesian Network called Maximum a Posterior (MAP) technique.

| Pr(Ft = T) | Pr(Ft = F) | Ft | Fi | Pr(OL = T|Ft, Fi) | Pr(OL = F|Ft, Fi) |
|------------|------------|----|----|-------------------|-------------------|
| 0.00861    | 0.99139    | T  | T  | 0.50000           | 0.50000           |
| Pr(Ft = T) | Pr(Ft = F) | T  | F  | 0.30488           | 0.69512           |
| 0.01764    | 0.98236    | F  | T  | 0.00588           | 0.99412           |
|            |            | F  | F  | 0.03885           | 0.96115           |

| OL | Pr(P = T|OL) | Pr(P = F|OL) |
|----|------------|------------|
| T  | 0.00451    | 0.99549    |
| F  | 0.00451    | 0.99549    |

| OL | Pr(Si = T|OL) | Pr(Si = F|OL) |
|----|------------|------------|
| T  | 0.00505    | 0.99495    |
| F  | 0.00021    | 0.99979    |

Table 1. Conditional Probability Table (CPT) of the Bayesian network in Figure 1

From Bayesian Network in figure 1 and the CPT in table 1 we can determine the probability of a proposition given some other propositions by applying Prolog rules presented in [12]. Therefore, we can answer the following problems:

- What is Pr (Si = T | Ft = T)? (i.e., what is the probability of static inverter disruption happens when fallen tree disruption also happens?)
- What is Pr(Ft = T, P = T)? (i.e., what is the probability of both fallen tree and pantograph disruptions happen?)
- What is Pr(Ft = T | P = T, Fi = F)? (i.e., what is the probability of fallen tree disruption happens when pantograph disruption also happens but fire disruption does not?)

To simplify the notations, for any node Ni of random variable, we abbreviate the standard notations with simplified logic representations as in table 2.
Standard Notation | Simplified Logic Notation
--- | ---
\(\Pr(N_1 = T \mid N_2 = T)\) | \(\Pr(N_1 \mid N_2)\)
\(\Pr(N_1 = T \mid N_2 = F)\) | \(\Pr(N_1 \mid \neg N_2)\)
\(\Pr(N_1 = T \mid N_2 = T, N_3 = T)\) | \(\Pr(N_1 \mid N_2 \land N_3)\)
\(\Pr(N_1 = T \mid N_2 = T, N_3 = F)\) | \(\Pr(N_1 \mid N_2 \land \neg N_3)\)
\(\Pr(N_1 = T \mid N_2 = F, N_3 = T)\) | \(\Pr(N_1 \mid \neg N_2 \lor N_3)\)

| Table 2. Simplified logic notations for probability problems. |

Prolog determines the answer to any probabilistic query by recursively applying the following rules:

- **Probability of conjunction:**
  \[ \Pr(X_1 \land X_2 \mid Cond) = \Pr(X_1 \mid Cond) \cdot \Pr(X_2 \mid X_1 \land Cond). \]  
- **Probability of certain event:**
  \[ \Pr(X \mid Y_1 \land ... \land X \land ...) = 1. \]  
- **Probability of impossible event:**
  \[ \Pr(X \mid Y_1 \land ... \land \neg X \land ...) = 0. \]  
- **Probability of negation:**
  \[ \Pr(\neg X \mid Cond) = 1 - \Pr(X \mid Cond). \]  
- **If condition involves a descendant of \( X \), then Bayes’ theorem is used:**
  If \( Cond_0 = Y \land Cond \) where \( Y \) is a descendant of \( X \) in the Bayesian network, then
  \[ \Pr(X \mid Cond_0) = \frac{\Pr(X \mid Cond) \cdot \Pr(Y \land X \land Cond)}{\Pr(Y \mid Cond)}. \]  
- **Cases when condition \( Cond \) does not involve a descendant of \( X \):**
  - If \( X \) has no parent then \( \Pr(X \mid Cond) = \Pr(X) \), the value of \( \Pr(X) \) is given.
  - If \( X \) has parents \( Parents \), then
    \[ \Pr(X \mid Cond) = \sum_{S \in \text{possible states}(Parents)} \Pr(X \mid S) \cdot \Pr(S \mid Cond). \]

As an example, the answer for the problem: “what is the probability of fallen tree (\( Ft \)) occurs given the occurrence of static inverter (\( Si \))?” is derived using rule [1]–[6] until all the rules are satisfied; we have \( \Pr(Ft \mid Si) = 0.03549 \).

3. **Constructing Bayesian Network for disruption patterns**

3.1. **Disruption categorization**

To construct a reliable Bayesian network for disruption patterns, firstly we collect the data of train system disruptions from the official account of PT. *Kereta Commuter Indonesia* (the account @commuterline), which occurred between September 2014 and December 2017. This time interval selection is performed to improve the accuracy of disruption probabilities. The disruptions mostly occurred in 2015 and slowly decreased until 2017 [2].

After the disruption data collection, we categorize tweets using regular expression to standardize disruption categories. The tweets categorization resulted in 22 disruption categories, namely:
3.2. Part-whole relation of train system

Part-whole relation is a fact that explains about a relationship between a part of object with the whole object itself. The part-whole relation is used to determine the structure of the Bayesian network as well as to ensure that every edge in the network accurately represents cause and effect relationship between two disruptions. In this research, we obtain the part-whole relation of the train system by interviewing the officer of PT. Kereta Commuter Indonesia. From the interview we have the part-whole relation of train system descriptions as follows:

- A pantograph disruption may trigger an overhead line disruption, and vice versa.
- Suspension disruption is an independent disruption, it does not lead to or triggered by other disruptions.
- Compressor is a part of train system that has a function to produce compressed air that is almost always used for brakes, horn, and sometimes for powering train doors. Brakes, horn, and doors disruptions also may lead to series disruption.
- Traction is a train driver. If there is a traction disruption, the series cannot work.
- MG disruption is not triggered by other disruptions. However, if there is MG disruption, the train system may not work and it can lead to series disruption.
- Railway disruption is an independent disruption, it does not lead to or triggered by other disruptions.
- Service table disruption may lead to switch or signal disruption. It also takes responsibility for making a train track when it is about to enter the station.
- The signal disruption can lead to switch disruption, and vice versa.
- Static inverter disruption is triggered by overhead line disruption. It is a part of train that has a function to convert the direct current from the overhead line into alternating current for the air conditioners inside the train. Air conditioner (AC) disruption also can lead to series disruption.
- Emergency or also known as dead man pedal disruption is an independent disruption, it does not lead to or triggered by other disruptions.
- When fallen tree disruption occurs, it can lead to overhead line disruption. The fallen tree can affect the overhead line performance.

From those part-whole relations, we can construct part-whole relation graph as in Figure 2.
3.3. **Bayesian Network construction**

There are some methods to obtain Bayesian network structure [15]. In this research, we construct Bayesian network structure from part-whole relation of train system by eliminating cycle into Bayesian networks structure. We did not evaluate the structure further, since working this out in detail is beyond the scope of this paper. From the aforementioned part-whole relation we obtain four Bayesian Network as in figure 3, 4, 5 and 6.

**Figure 2.** A graph representing the part-whole relation of a train system.

**Figure 3.** Bayesian Network 1, which is obtained from figure 2 by removing the edges from switch to signal and pantograph to overhead line.

**Figure 4.** Bayesian Network 2, which is obtained from figure 2 by removing the edges from signal to switch and pantograph to overhead line.
Figure 5. Bayesian Network 3, which is obtained from figure 2 by removing the edges from signal to switch and overhead line to pantograph.

Figure 6. Bayesian Network 4, which is obtained from figure 2 by removing the edges from switch to signal and overhead line to pantograph.

For each Bayesian network structure, we need to construct the CPT of the network [16]. We construct the CPT for all Bayesian networks with Maximum a Posterior (MAP) technique to avoid zero division when we reason about the probability of the disruptions. The full CPT of all Bayesian networks can be accessed on [17].

4. Reasoning disruption patterns with Prolog
There are three basic constructions of Prolog programs: facts, rules, and queries. Before implementing the model, we also need to specify the queries, so we can conveniently determine the Prolog rules. The list of queries are obtained from analyzing previous research in [3], our interview with Commuter Line’s officer, as well as our observations. These queries are specified as follows:

Q1. Calculate the probability of a certain disruption given other disruptions.
Q2. Find paths from one disruption to other disruptions from a given conditional probability problem.
Q3. Find disruption path from a disruption to another disruption that contains and does not contain certain disruption.
Q4. Find the common triggered disruptions from two other disruptions. This query aims to find disruptions that directly led by two other disruptions. For any disruption $D_1$, $D_1$ is the common triggered disruption of $D_2$ and $D_3$ if both $D_2$ and $D_3$ have direct edges to $D_1$. 

Q5. Find the common triggering disruptions that lead to other disruptions. This query aims to find disruptions that may directly trigger two other disruptions. For any disruption \( D_1 \), \( D_1 \) is the common triggering disruption of \( D_2 \) and \( D_3 \) if \( D_1 \) has a direct edge to \( D_2 \) and \( D_3 \).

Q6. Find the most triggering disruption in the network.

Q7. Find the most triggered disruption in the network.

After specifying the queries, we need to determine the Prolog facts. The facts have a role as a knowledge base in the reasoning process. We put facts about part-whole relation of the train system and Bayesian network structure, including the information in the CPT. The Prolog representation of the facts are quite simple and straightforward and it can be accessed on [17].

Hereafter, we need to construct Prolog rules, so we can answer the queries Q1–Q7. These rules are constructed using the previously defined facts in the knowledge base. We define Prolog rules for all the queries as follows:

1. For Q1 we use Prolog rules in [12] which implement rules (1), (2), (3), (4), (5), and (6) in Section 2.

2. For Q2 we have the following Prolog rule:

\[
\text{adjacent}(A, B, [A,B]) := \text{parent}(A, B).
\]

\[
\text{path}(A, B, [A,B]) := \text{adjacent}(A, B, [A,B])
\]

\[
\text{path}(A, B, [A \mid \text{Rest}]) :=
\]

\[
\text{adjacent}(A, X, [A,X]),
\]

\[
\text{path}(X, B, \text{Rest}).
\]

\[
\text{path\_probability}(A, B, \text{Path}, \text{Prob}) :=
\]

\[
\text{path}(A, B, \text{Path}), \text{prob}(B, [A], P),
\]

\[
\text{Prob is P}.
\]

3. For Q3 we have the following Prolog rule:

\[
\text{show\_path}(A, A, T, P) := \text{reverse}([A \mid T], P).
\]

\[
\text{show\_path}(A, Z, T, P) := \text{edge}(A,B), \text{not}\ (\text{member}(A, T)), \text{show\_path}(B, Z, [A\mid T], P).
\]

\[
\text{show\_path}(A, B, P) := \text{show\_path}(A, B, [], P).
\]

\[
\text{causing}(A, B, \text{Path}, \text{Cause}) :=
\]

\[
\text{show\_path}(A, B, \text{Path}), \text{subset}([\text{Cause}], \text{Path}).
\]

\[
\text{match}(L1, L2) := \text{member}(X, L1), \text{member}(X, L2).
\]

\[
\text{avoid}(L1, L2) := \text{not}\ (\text{match}(L1, L2)).
\]

\[
\text{not\Causing}(A, B, \text{Path}, \text{Avoid}) :=
\]

\[
\text{show\_path}(A, B, \text{Path}), \text{avoid}([\text{Avoid}], \text{Path}).
\]

4. For Q4 we have the following Prolog rule:

\[
\text{com\_triged\_dis}(X, Y, Z) :=
\]

\[
\text{ancestor}(X, Z), \text{ancestor}(Y, Z),
\]

\[
\text{show\_com\_triged\_dis}(X, Y, \text{Com\_Triged\_Dis}) :=
\]

\[
\text{findall}(Z, \text{com\_triged\_dis}(X, Y, Z), L), \text{sort}(L, \text{Com\_Triged\_Dis}).
\]

5. For Q5 we have the following Prolog rule:

\[
\text{com\_triging\_dis}(X, Y, Z) :=
\]

\[
\text{ancestor}(Z, X), \text{ancestor}(Z, Y), X \leq Y.
\]

\[
\text{show\_com\_triging\_dis}(Z, \text{Com\_Triging\_Dis}) :=
\]

\[
\text{findall}([X,Y], \text{com\_triging\_dis}(X, Y, Z), L), \text{sort}(L, \text{Com\_Triging\_Dis}).
\]

6. For Q6 we have the following Prolog rule:
descendant(X,Y) :- parent(X,Y).
descendant(X,Y) :- parent(X,Z), descendant(Z,Y).

descendants([Node, Descendant, Length]) :-
    node(Node), findall(A, descendant(Node,A), L),
    sort(L, Descendant), length(Descendant, Length).

all_descendant(Result) :-
    findall([Node, Descendant, Length],
    descendants([Node, Descendant, Length]), Result).

compare_descending('>', [_, _, X], [_, _, Y]) :- X > Y, !.

7. For Q7 we have the following Prolog rule:
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
ancestors([Node, Ancestor, Length]) :-
    node(Node), findall(A, descendant(A, Node), L),
    sort(L, Ancestor), length(Ancestor, Length).

all_ancestor(Result) :-
    findall([Node, Ancestor, Length], ancestors([Node, Ancestor, Length]), Result).

sort_ancestor_descending(Sorted) :-
    all_ancestor(Result),
    predsort(compare_descending, Result, Sorted).

max_ancestor(L) :-
    sort_ancestor_descending(Result), nth0(0, Result, L).

detail_max_ancestor([Node, Ancestor, Length]) :-
    max_ancestor(L),
    nth0(0, L, Node),
    nth0(1, L, Ancestor),
    nth0(2, L, Length).

The main menu of our Prolog program is depicted in figure 7 and it is accessible on 17.
To reason about a particular Bayesian network n (where n ∈ {1,2,3,4}), one can consult the following files consecutively: rules.pl, partwholerelation.pl, and bayesian_networkn.pl.
5. Query implementation: result and discussion

This research deals with knowledge representation formalism and automated reasoning, as one of the fields of artificial intelligence. Therefore, unlike empirical research, we evaluate the accuracy of our approach by generating traces from knowledge base construction in Section 3 and Prolog reasoning in Section 4 using queries.

Here we show the query implementation of our program. To make our description brief, we only show a Prolog query for the first Bayesian network in figure 3. To ensure the program gives appropriate responses, we also create some test cases for testing the program functionality. The complete test cases and their corresponding results are available on [17].

We do not compare our results empirically with other outcomes of other approaches because we focus on the formal reasoning of the disruption patterns and not on the quantitative outcomes of the reasoning. One of our primary objectives is to build a Prolog-based expert system for reasoning about probability when a disruption happened and show the causative relationship among the disruptions. Hence in the future, PT. Kereta Commuter Indonesia can use the expert system to predict the probability of a disruption and trace the cause and effect when a disruption happened. Examples of query implementations of Q1–Q7 are described as follows.

(1.) An example of the query implementation of Q1 is depicted in figure 8.

From query shown in figure 8, we obtain the probability if fallen_tree disruption happened.
given fire disruption happened is 0.00861. This number is obtained from rule 6 in Section 2.

(2.) An example of the query implementation of Q2 is depicted in figure 9.

Figure 9. An example of the query implementation of Q2 to calculate Pr(series | fire).

From query shown in figure 9 we obtain a disruption path from fire disruption to series, namely fire → overhead_line → static_inverter → ac → series, with the probability Pr(series|fire) = 0.08666. This number is obtained from the rules that explained in Section 2.

(3.) An example of the query implementation Q3 is depicted in figure 10.

Figure 10. Reasoning about disruption path, a path that contains a particular disruption, and a path that does not contain a particular disruption.

From query shown in figure 10 we obtain that there is a disruption path when series disruption happened and caused by compressor disruption. Brake disruption may become one of the intermediate disruptions. However, there are also disruptions path from compressor disruption to the series disruption that does not contain door disruption in between.

(4.) An example of the query implementation of Q4 is depicted in figure 11.
From the query shown in figure 11, we reason that fallen tree and fire disruptions can lead to other disruptions, namely: AC, overhead line, pantograph, series, and static inverter.

(5.) An example of the query implementation of Q5 is depicted in figure 12.

From the query that shown in figure 12, we obtain that overhead line and pantograph disruption is the common triggering of fallen tree and fire disruption.

(6.) An example of the query implementation of Q6 is depicted in figure 13.

From the query shown in figure 13, we reason that the most triggering disruption (using Bayesian network 1 in figure 3) is fallen tree disruption. It can lead to five other disruptions, namely: AC, overhead line, pantograph, static inverter, and series disruption.

(7.) An example of the query implementation of Q7 is depicted in figure 13.
From the query shown in Figure 14, we reason that the most triggered disruption (using Bayesian network 1 in Figure 3) is series disruption. It can be triggered by 12 other disruptions, namely: ac, break, compressor, door, fallen tree, fire, horn, overhead line, speedometer, static inverter, traction, and wiper disruption.

6. Conclusion and future works
In this paper, we have constructed a formal model to reason about the disruption patterns of train system by combining Bayesian network and Prolog. The proposed system can be used to reason cases related to data patterns and part-whole relation in train system. We pose several queries to all of Bayesian networks regarding the probability, causality between each disruption to other disruptions, and the most triggering and triggered disruptions in each network. We show that Prolog can easily and effectively be used to reason about probability in Bayesian network. We exploit Prolog capability to generate a general rule to reason about probability in Bayesian network. For any other Bayesian network, we only need to change the knowledge base of the program.

We also conclude that constructing a Prolog-based expert system for reasoning the disruption patterns in train system is an achievable work. We use Prolog program to determine the probability of a disruption, the conditional probability of disruptions, the causative path between two disruptions, the common triggered and triggering disruptions, and other possible relationships derived from the Bayesian network. Moreover, for all of the Bayesian networks in Figure 3, 4, 5, and 6, we obtain that the most triggering and triggered disruptions are fallen tree and series disruptions, respectively.

There are some improvements that can be conducted in the future works. Constructing disruption patterns of train system using structure learning in Bayesian network is needed to improve the accuracy of the disruptions’ probabilities. In addition, to guarantee the existing Bayesian network of the disruption patterns in train system is the most suitable one, evaluation process of the networks is definitely required. Moreover, we also still need to build more comprehensive Prolog rules for showing more details of the causative relationship among disruptions in train system.

References
[1] PT Kereta Commuter Indonesia 2017 Info Perusahaan accessed on October 19, 2017. URL http://www.krl.co.id/
[2] Purba D O 2018 PT. KCI Sebut Gangguan Teknis KRL Berkurang pada 2017 accessed on May 25, 2017. URL https://megapolitan.kompas.com/read/2018/01/04/19291001/pt-kci-sebut-gangguan-teknis-krl-berkurang-pada-2017
[3] Rusmawati Y and Rismala R 2016 Towards the Modelling of Dynamic Train Systems: Disruption Pattern
Analysis Information and Communication Technology (ICoICT), 2016 4th International Conference on (IEEE) pp 274–278

[4] Pearl J 1988 Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (Representation and Reasoning)

[5] Heckerman D, Mamdani A and Wellman M P 1995 Real-world Applications of Bayesian Networks, Communications of the ACM 38 24–26

[6] Cooper G F 1990 The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks, Artificial Intelligence 42 393–405

[7] Wellman M P, Breese J S and Goldman R P 1992 From Knowledge Bases to Decision Models, The Knowledge Engineering Review 7 35–53

[8] Kersting K and De Raedt L 2001 Towards Combining Inductive Logic Programming with Bayesian Networks International Conference on Inductive Logic Programming (Springer) pp 118–131

[9] Kersting K and De Raedt L 2007 Statistical Relational Learning 291

[10] Ngo L and Haddawy P 1997 Answering Queries from Context-sensitive Probabilistic Knowledge Bases, Theoretical Computer Science 171 147–177

[11] Gutmann B, Thor I and De Raedt L 2011 Learning the Parameters of Probabilistic Logic Programs from Interpretations Joint European Conference on Machine Learning and Knowledge Discovery in Databases (Springer) pp 581–596

[12] Bratko I 2001 Prolog Programming for Artificial Intelligence (Pearson education)

[13] Blackburn P, Bos J and Striegnitz K 2006 Learn Prolog Now! vol 7 (College Publications London)

[14] Bramer M 2005 Logic programming with Prolog vol 9 (Springer)

[15] Xing-Chen H, Zheng Q, Lei T and Li-Ping S 2007 Learning bayesian network structures with discrete particle swarm optimization algorithm Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on (IEEE) pp 47–52

[16] Koller D and Friedman N 2009 Probabilistic Graphical Models: Principles and Techniques (MIT press)

[17] Reasoning Train Disruption With Prolog URL https://github.com/yunitarp/ReasoningTrainDisruptionWithProlog/