Evaluating Bayesian Networks from Multi-Agent Simulations

Ludi van Leeuwen

July 10, 2022

Abstract

There are probabilistic methods for reasoning with evidence in court cases. Bayesian Networks have been suggested as one of these methods, but we do not know how suitable they would be in the domain of law, which is not traditionally a statistical field. This work investigates Bayesian Networks for criminal cases by creating multi-agent simulations of crimes, collecting frequency data on these simulations, using the K2 algorithm to generate Bayesian Networks from this data, and then evaluating the generated Bayesian Networks on structural, performance and human factor criteria. This is done for 3 simple simulations: one non-spatial simulation of a murder, one spatial simulation of a home-robbery with evidence, and one spatial simulation of a street robbery in a non-trivial environment.

We find that the Bayesian Networks are generally accurate and perform well even if we do not know the exact probabilities as generated by the simulation. However, it is implausible that human modellers could replicate the cpt's of the networks. Apart from that, private knowledge, conditioning on implicit factors and the effects of non-ideal operationalisation are factors that need to be solved before Bayesian Networks can be responsibly applied in court.

Contents

Chapter 1

Introduction

When we find evidence for a hypothesis that we have held in the back of our minds, our belief in the hypothesis increases. This is the basic idea behind reasoning with evidence. In a constellation of hypotheses and pieces of evidence, we want to construct a network that will lead us to believe as many true hypotheses as possible, given the evidence that we have.

However, evidence itself is elusive, and its connection to hypotheses is as well - how can we be sure that some evidence supports some hypothesis? Even if we know that it does, how can we express how strong the piece of evidence is? Some evidence is very weak, and only after a tedious process of ruling out other factors and careful investigation and collection of other pieces of evidence, we can come to a conclusion about a hypothesis. On the other hand, some evidence is so strong that it leaves no room for doubt.

We all have intuitions about evidence strength - but can we make these intuitions precise? Additionally, can we manage the complex realities of weak evidence for many different hypotheses?

1.1 Bayes

We have a way of expressing how we should use evidence to update our beliefs in hypotheses. We can do this with the use of probabilities. When we reason probabilistically with evidence, we want to use the gold standard - Bayes Law. Let's say that we find some piece of evidence e, we want to know how our finding e affects our belief in some hypothesis h. We can express how much we believe h given e, using Bayes Law:

$$P(h|e) = \frac{P(e|h) \cdot P(h)}{P(e)}$$

This is the simplest form of Bayesian reasoning, one piece of evidence, and one hypothesis. But real life is not so simple, and we are constantly reasoning with many pieces of evidence. Small hypotheses can serve as stepping stones to greater hypotheses. One way of handling the complexity of interacting pieces of evidence and hypotheses is by using Bayesian Networks.

A Bayesian Network is a directed, acyclic graph that represents the joint probability distribution over a set of events (**?**). BNs have nodes and arcs. Formally, the nodes are random variables that represent events. In case of our Bayesian Networks, these random variables represent hypotheses and pieces of evidence. An arc represent a conditional relationships between two nodes. Conditional probability is expressed in the conditional probability table (cpt) of every node. A node that is not conditionally dependent on any other node (has no incoming arcs) is independent.

An example of a Bayesian Network can be seen in Figure 1.1. It expresses a simple reasoning step: how should our belief in the hypothesis that our train will be late in Groningen, change once we learn about the train station renovations in Zwolle?

In the figure, the node 'train_on_time_in_groningen' is a hypothesis. Nodes with one or more incoming arcs are conditionally dependent on their parent node(s). In our example network, the node 'station_renovation_zwolle' is conditionally dependent on its parent. The links between the nodes are links of conditional probability, and do not correspond to real-world causality, although they are sometimes interpreted as being causal (**?**).

We can reason with Bayesian Networks by setting evidence on one or more nodes that we can observe - our evidence. This means that we say whether some observable node is true or false, and use that information to update the network. We can now find a new posterior probability for a node that we cannot observe directly (yet). In Figure 1.1, this means that when we learn that there are station renovations going on at Zwolle, the our belief that our train will be late in Groningen increases from a probability of 0.01, to a probability of 0.8.

1.2 This project

Using Bayesian Networks, we can make explicit how we think that we should reason about hypotheses given pieces of evidence. In our toy example, this is not very consequential, it is a very simple network. However, Bayesian reasoning and Bayesian Networks specifically, can be applied to many domains. One of which is AI and law, specifically, Bayesian Networks might be tools that could be useful in court cases, by organising evidence and hypotheses. A Bayesian Network might make explicit how reasoners should update their beliefs based on the evidence, which might increase transparency and fairness. In the ideal case, a Bayesian Network might stop a judge from making a reasoning error!

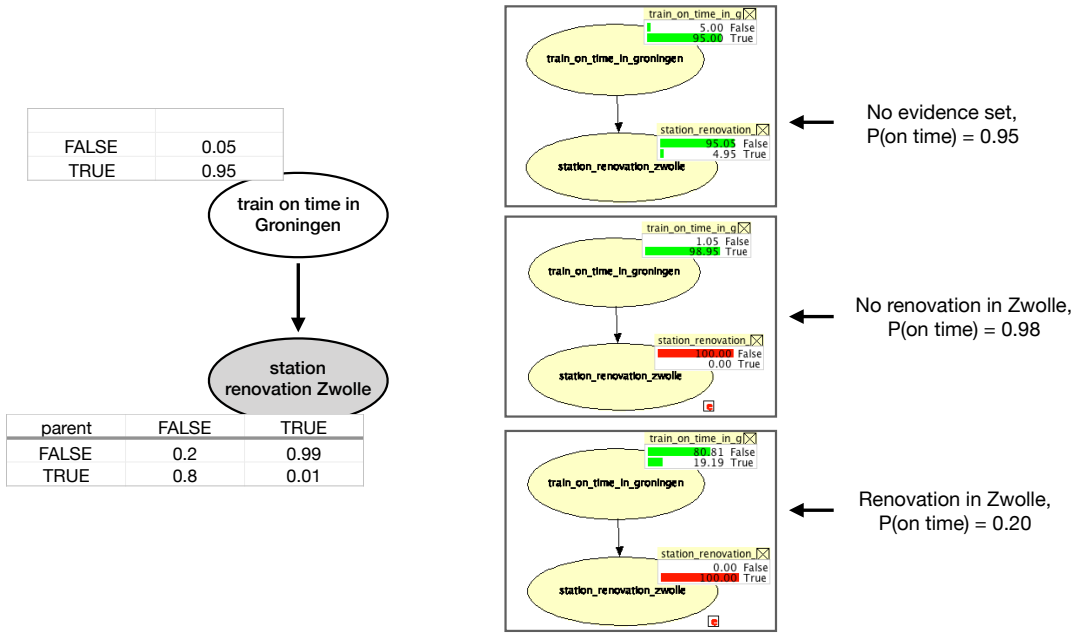

Figure 1.1: An example of a Bayesian Network, with and without evidence set.

However, creating Bayesian Networks is very complex, especially in domains where probabilistic data is sparse. We also do not have a good idea if our methods for creating Bayesian Networks in criminal situations work, since it is difficult to test these networks empirically - you cannot ask a murderer to murder again, for the statistics. This is why we need a grounding to investigate whether our methods for building Bayesian Networks actually work.

This is why this project proposes to ground Bayesian Networks for crimes in multi-agent simulations of crimes. We can model crime cases using simulations to a level of realism we desire. We can see what probabilities emerge out of these simulations by running them multiple times, and see if we can capture this probabilistic information into Bayesian Networks. We can collect exactly the frequency information that we need. This means that we can focus on assessing the Bayesian Networks, since we know what frequency information they should replicate.

1.2.1 Research Questions

- **Can we create Bayesian Networks that correctly reflect multi-agent simulations of crimes?**

- **If we can, under what assumptions do these networks function?**

1.3 Overview of the thesis

Chapter 2 is an overview of the state of the art, Chapter 3 proposes a pipeline for creating simulations and automatic Bayesian Networks, and evaluation criteria that state when a Bayesian Network correctly reflects multi-agent simulations of crimes. Chapter 4, 5 and 6 are applications of this pipeline in three separate simulations: first a simple, non-spatial simulation based on (**?**), a simple spatial simulation of a home-robbery, and finally a spatial simulation of a street-robbery. For each of these simulations, some simple experiments are done to investigate several problematic aspects of Bayesian Networks. These simulations are then each evaluated based on the evaluation criteria set out in Chapter 3. Finally, we draw conclusions in Chapter 7.

You can interact with the simulation in Chapter 6 on https://shielded-journey-34533.herokuapp.com.

Code for this project can be found at https://github.com/aludi/simulationTest.

Chapter 2

State of the Art

We want to create grounded Bayesian Networks for reasoning with evidence. This can be done by creating multi-agent simulations, and building Bayesian Networks based on the events in these simulations. In this section, the relevant background is laid out on reasoning with evidence, Bayesian Networks, and multi-agent simulations.

2.1 Reasoning with evidence

We consider three main directions in the field of reasoning with evidence within law (**?**). The first direction is via argumentation approaches, where hypotheses and evidence are represented as propositions that attack or support each other. The second direction is via scenario approaches, where more-or-less coherent hypotheses are combined into stories (**?**), (**?**), which are supported with evidence. The third direction is via probabilistic approaches, where hypotheses and evidence are assigned probabilities, and the relation between hypothesis and evidence is represented with conditional probability. There are also be hybrids that combine or synthesise aspects of each approach, see, for example: (**?**) and (**?**). This project does not consider the argumentative approach, and is mainly based on the work of Vlek (**?**), (**?**) and Fenton (**?**), (**?**). The networks in these papers describe whole criminal situations (scenarios), but use Bayesian Networks and hence have a probabilistic component.

2.2 Bayes and Bayesian Networks

We have already seen the power of Bayes's law in the introduction. It tells us how much we have to change our belief in a hypothesis once we find a piece of evidence. To do this, we need to have probabilistic or frequentist information about the likelihood ratio and the

prior.

Bayesian reasoning, without Bayesian Networks, have been used by Dahlman in 'event trees', specifying different branches of combinations and their probabilities (**?**).

However, just using Bayes law is very difficult, because sometimes we want to condition on multiple pieces of evidence. Doing all the calculations is tedious, hence we use the computational tools called Bayesian Networks.

Bayesian Networks can represent aspects of a criminal case or can attempt a scenario-like hybrid and represent the entire case - modelling actual crime cases (**?**), (**?**), cases from fiction (**?**) or fictionalized crime cases (**?**). In situations where the network represents aspects, the nets model DNA or blood-spatter evidence, for methods see (**?**).

Bayesian Networks can be built by hand, by experts or academics, in (proprietary) software like AgenaRisk or Hugin. They can also be built by hand in PyAgrum (**?**), a free Python software package, which does not have a GUI but has everything else. In this project, PyAgrum was used. Alternatively, Bayesian Networks can be automatically constructed from large datasets - PyAgrum also offers the opportunity for that. Automated Bayesian Network building is not plausible in the legal-evidence domain, because the data that we need is notoriously sparse.

2.2.1 Evaluating Bayesian Networks

Bayesian Networks are a promising tool, but there are a lot of open questions:

1. The use of Bayesian Networks is not straightforward, neither for the builder or for the interpreter. From **?**: it is complex, time-consuming, hard to explain, and, the 'repeatability [...] leaves much to be desired. Node definitions and model structures are often directed by personal habits, resulting in different models for the same problem, depending on the expert'.

2. The granularity of the network - how do we know which nodes hypotheses and pieces of evidence to include? Ideally we would model as detailed as possible, but as we increase the number of nodes, we increase the complexity, which increases the probability of mistakes, and the time spend on the network.

3. The links: how do we know which events depend on each other?

4. The numbers - there's not just subjectivity in selecting the nodes, and drawing the links, but the probabilities that we have to fill in into the cpt themselves are the most obvious stumbling block. We can identify that there has to be some sort of correlation between 'smoke' and 'fire', but how to express this in numbers? Fenton argues that we're just making something explicit that we would otherwise have left implicit. But what are the consequences of making something explicit, but doing it wrongly? How

robust is the network against imprecise or wrong frequencies? If we use frequencies (or subjective probabilities based on frequencies), how do we decide what set of events is included when we start to count (this is the problem of the reference class, see (**?**), (**?**)). Probabilities can be pure frequencies, or can be subjectively elicited from experts (**?**), (**?**).

This project cannot address all of these problems, because it's too much. My main focus will be on problem 4. As we mentioned before, one of the problems that can plague Bayesian Network creation is the lack of well-defined and plentiful data. But what if we would have this data about criminal cases? Then we could test our methods for Bayesian Networks without worrying about the subjectivity and lack of data of the numbers in the cpt. If we can create a grounded 'data-generating' environments for criminal cases, we can test our Bayesian Networks against them.

2.3 Agent simulations

We can create such a data-generating environment for criminal cases by the use of multi-agent simulations. In a multi-agent simulation, agents observe and interact with their environment. The environment and the agent behaviour is fully controlled by its programming and all randomness can be accounted for. This means that we know exactly with which frequencies fully-specified events occur. Running the simulation multiple times generates a lot of data, which will serve as input to automatically build a Bayesian Network from data. This means we use a multi-agent simulations as a ground for our theory-testing. In this project, the simulations will be programmed in Python using the MESA framework (**?**).

Multi-agent simulations have been used to investigate the criminal domain and to test out sociological theories. By creating spatially explicit simulations, the complex interactions of agents with their locations can be represented better than in traditional models - these agent-based models can model criminal hotspots (**?**), theories of behaviour (**?**), and police strategies in urban crime (**?**). Weaknesses identified with multi-agent models for criminological research are insufficiently complex models, unclarity on the effect of temporal resolution, and lack of a systemic approach (**?**).

2.4 Conclusion

The two main ideas are Bayesian reasoning, specifically as implemented in Bayesian Networks, and computational simulations of agents. The simulation is supposed to be the grounding for the Bayesian Networks.

But why do we want to use Bayesian Networks in the first place? In the best possible case, a Bayesian Network would help us to make the correct reasoning steps, telling us

exactly how to weigh each piece of evidence in the grand scale of the simulated crime. This is a normative approach - it's telling us how to reason, but it is not (and not meant to be) an empirical approach. Bayesian networks are not a reflection of 'how people actually reason' - If you open up our minds, you won't see Bayesian Networks in there. One of the problems for normative models in decision making (**?**), is that we can only test how well our normative models work, if we already know what a good outcome would be. In a sense, we're doing experimental model testing for normative Bayesian Networks in this project.

Chapter 3

Method

3.1 Introduction

This chapter lays out the general method for creating agent-based simulations with automatic Bayesian Networks, as well as how to evaluate these Bayesian Networks.

We can consider an approach in four stages. First, we need a scenario, or set of alternative scenarios, that are to be modelled. These scenarios are written description of crime scenarios that contain the hypothetical events and evidence for these events, as postulated by the prosecutor and defence.

Second, based on this written description of the scenario(s) to be modelled, an agent-based simulation is created. In the simulation, all events of the scenario need to represented. The simulation can be as granular as desired. The simulation can be run multiple times, so that we can gain an understanding of possible underlying frequency information on the events in the scenario. The frequencies by which events occur in the simulation are collected.

Third, the collected frequencies are then used to automatically build a Bayesian Network on the simulation, using the K2 algorithm. Finally, the generated Bayesian Network is then evaluated based the criteria set out at the end of this chapter.

This four-step process will be explained in this chapter, and illustrated with a running example of a stabbing.

3.2 A Scenario

We start our process with one or more written scenarios. These scenarios can, in the first instance, be obtained from (abridged) court case descriptions. The scenarios should contain all and only those hypothesised events, and the evidence for such events, that are

relevant to the case. Both the prosecution and the defence should be able to select relevant events and their evidence.

Example 1 *Here we will introduce our running example, loosely based on the first scenario in (**?**): Mark and Jane live together. One day, a neighbour overhears that they've started to fight in the kitchen. Jane grabs a knife, and stabs Mark. Mark dies of the stab wounds, as confirmed by the forensic scientist at the scene. Jane's fingerprints are on the knife.*

3.3 An Agent-based Simulation

We need a way to transform the written text descriptions of the events in the scenario to observable events in the simulation. First we identify all relevant actors, objects and places in the scenario. Then we select all relevant events from the written description and attempt to operationalise them in the simulation. This means that we create a way to measure if the events take place within the simulation. This is done by means of 'reporters', or random variables. These reporters report whether a given event has taken place within the simulation.

The process of operationalisation is very subjective and depends entirely on the modeller's assumptions and their time, knowledge or skill constraints. The behaviour of agents, as reflections of actual people, can be modelled in a way that reflect real sociological theories of behaviour (such as in **?**), or with simple rules and a random number generator (such as here). The environment of the simulation can reflect a real-world geography with different affordances, or can be simplified. How a given event is operationalised is essential for the validity of the network: if it is unclear, or disputed, when an event is true in a simulation, the network should not be used.

Example 2 *We have modelled Jane and Mark as two agents in a simulation (Figure). The only 'psychological' aspect that these agents have, is an anger level, which has 4 escalating states = {not angry, fighting, grabbing knife, stabbing}. Jane is quicker to increase to the next anger level.*

jane_and_mark_fight *Both Mark and Jane have an anger level that is higher than 'not angry'*

jane_has_knife *Jane has an anger level of 'grabbing knife' and moved to the location of the knife*

jane_stabs_mark_with_knife *Jane has an anger level of 'stabbing', has the knife, and moved to Mark's position*

mark_dies *Jane has stabbed Mark, and if a random number generator (uniform, 0, 100) produces a number > 60*

E_neighbour *Either Mark and Jane have an anger level that is higher than 'not angry'*

E_prints *True when Jane has knife*

E_stab_wounds *True when Jane stabs Mark*

E_forensic *True when Mark dies*

Then, the operationalisation itself: In the simulation, certain events can be brought about. In our example, this could be the events of 'jane_stabs_mark', or 'E_forensic'. We need a way to observe these states: this is where reporters come in. A reporter is a random variable that is reports the outcome of a relevant event in the simulation and is embedded in the code. If an event happens (or does not happen), the reporter reports that the event is true (or false). In essence, the reporter (R) is a random variable (RV) (for a further explanation of Random Variables, see Chapter 8). In short, a random variable maps an event (e) to a truth value:

$$R : e \rightarrow \{0, 1\}$$

Over which events we define reporters, depends on which events we deem relevant. We could, in theory, create a combinatorial explosive number of reporters for any possible event, like $jane\_in\_position\_(0, 0)$, $jane\_in\_position\_(1, 0)$, etc, but pragmatically, these very granular reporters will not help us in modelling the case. This means that we need to find a balance between over-specification and under-specification.

A reporter's value is 0 by default, but maps an event to 1 whenever it happens in the simulation. When we run the simulation in our example, we expect to see that, whenever Jane and Mark are fighting, R : 'jane_and_mark_fight' $\rightarrow 1$. At a later time-step, if Jane stabs Mark, we would see: R : 'jane_stabs_mark' $\rightarrow 1$.

No matter how many reporters we define, we can combine all reporters at the end of one run of the simulation into a global state G.

$$G = (e_0 \rightarrow \{0, 1\} \times e_1 \rightarrow \{0, 1\} \times ... \times e_n \rightarrow \{0, 1\})$$

or, for n reporters:

$$G = R_1 \times R_2 \times ... \times R_n$$

A global state that would represent that every one of the postulated events and evidence happened, except that Mark did not die of the stabbing, and that there was no forensic

scientist to determine that he died of the stab wounds, would be represented as (reporters in the same order as in the listing above):

$$1, 1, 1, 0, 1, 1, 1, 0$$

Then, we collect these global states over the number of runs that we do for each experiment, which results in the output O of this stage of the method, is a series of global states, one for each run:

$$O = (G_0, G_1, ...G_{runs})$$

3.4 Creating a Bayesian Network from a Simulation Automatically

The output of an experiment is the collection of runs O, where each run is the global state G of the simulation, as measured by the random variables R. The reporters in the simulation are random variables. These reporters become the nodes in the Bayesian Network.

The Bayesian Network is generated automatically, using the automated BN learner method as implemented in pyAgrum. There are several learners implemented in pyAgrum.[1] The algorithm used in this experiment to structure the Bayesian Network from the simulation data is the K2 algorithm (**?**).

The K2 algorithm is a greedy search algorithm that attempts to maximise the posterior probability of the network by correctly connecting parent nodes to child nodes. It takes an ordering on the nodes as input. The first node in the ordering X_0 does not have any parents. For two nodes X_i and X_j, X_j cannot be the parent node of X_i if $j > i$: a node can only have a parent that is earlier in the ordering. The algorithm processes each node X_i by adding possible parent nodes to X_i (as constrained by the ordering), and maximising the score of the network. The algorithm stops if there are no parents to add, no parents improve the score, or the node has reached the maximal number of parents (**?**).

Due to this ordering constraint, the K2 algorithm is efficient. However, finding the ordering of the nodes as input for the network is not trivial. The ordering should be meaningful, to reflect causal or temporal information present in the domain. Through our simulation, we can find the temporal ordering of the nodes by collecting the temporal order in which hypothesis events occur. The evidence nodes are added at the end, to ensure that evidence nodes are never parents to hypothesis nodes.

[1] An introduction to structure learning using PyAgrum: http://webia.lip6.fr/~phw/aGrUM/docs/last/notebooks/structuralLearning.ipynb.html

Example 3 *The temporal ordering of events in the simulation would be:*

{*'E_neighbour', 'jane_and_mark_fight', 'jane_has_knife', 'E_prints', 'jane_stabs_mark_with_knife', 'E_stab_wounds', 'mark_dies', 'E_forensic'* }

The ordering we want to give to K2 to save the temporal structure and the evidence-idiom **?***:*

{ *'jane_and_mark_fight', 'jane_has_knife', 'jane_stabs_mark_with_knife', 'mark_dies', 'E_neighbour', 'E_prints', 'E_stab_wounds', 'E_forensic'* }

The ordering is calculated as follows: at every run, we note down which hypothesis events happen in which order. We only note down the order, not the time-step at which an event occurs. We count how often each set of events occurs. We select the set of events that contains the most events, as this is likely the set that represents the entire 'causal' chain, or an entire scenario. If there are two sets of events of equal length, we pick the one that occurs most frequently. We add this set first to the ordering. We then look at the number of hypothesis nodes we have left, and add the largest set we can, until we have added all the hypothesis nodes to the ordering. Then we add the evidence nodes to the ordering.

For example, let's say we have 4 hypothesis nodes, $X_1, ...X_4$. In Run 1 we find X_2, X_3, X_4, in Run 2 we find X_1, in Run 3 X_2, X_3, X_4 again, but Run 4 is X_1, X_2, X_3. We select the longest sets first, and since X_2, X_3, X_4 occurs more often than the other one, we add it to the ordering first. The ordering is now X_2, X_3, X_4. We have 4 hypothesis nodes, so we need to add a set of length 1, which is X_1, which results in an ordering of X_2, X_3, X_4, X_1 that we add evidence to, and apply to the K2 algorithm.

Example 4 *We give the collection of global states O from our example to K2, as well as the temporal ordering as laid out in the previous example. This results in the a Bayesian Network that is shown in Figure 3.1.*

After the network is generated, we need to manually change a thing about it. The algorithm does not know how to handle impossible states, which are cases where combinations of valuations for parent nodes are impossible. For example, in Figure 3.1, the value of 'E_forensic'. 'E_forensic' has two parents: 'mark_dies', as well as 'E_neighbour'. In the simulation, it can never be the case that Mark dies, but the neighbour does not hear and report on fighting, or the combination of events $'mark\_dies' \to 1 \wedge 'E\_neighbour' \to 1$. However, the event for the thief stealing the object, can have both 'motive' and 'knowing about object' as parent nodes. The algorithm does not know how to deal with these situations and the resulting network will crash. This is why the probability for 'stealing' given the impossible combination of the parent state values, will be manually set to 0.

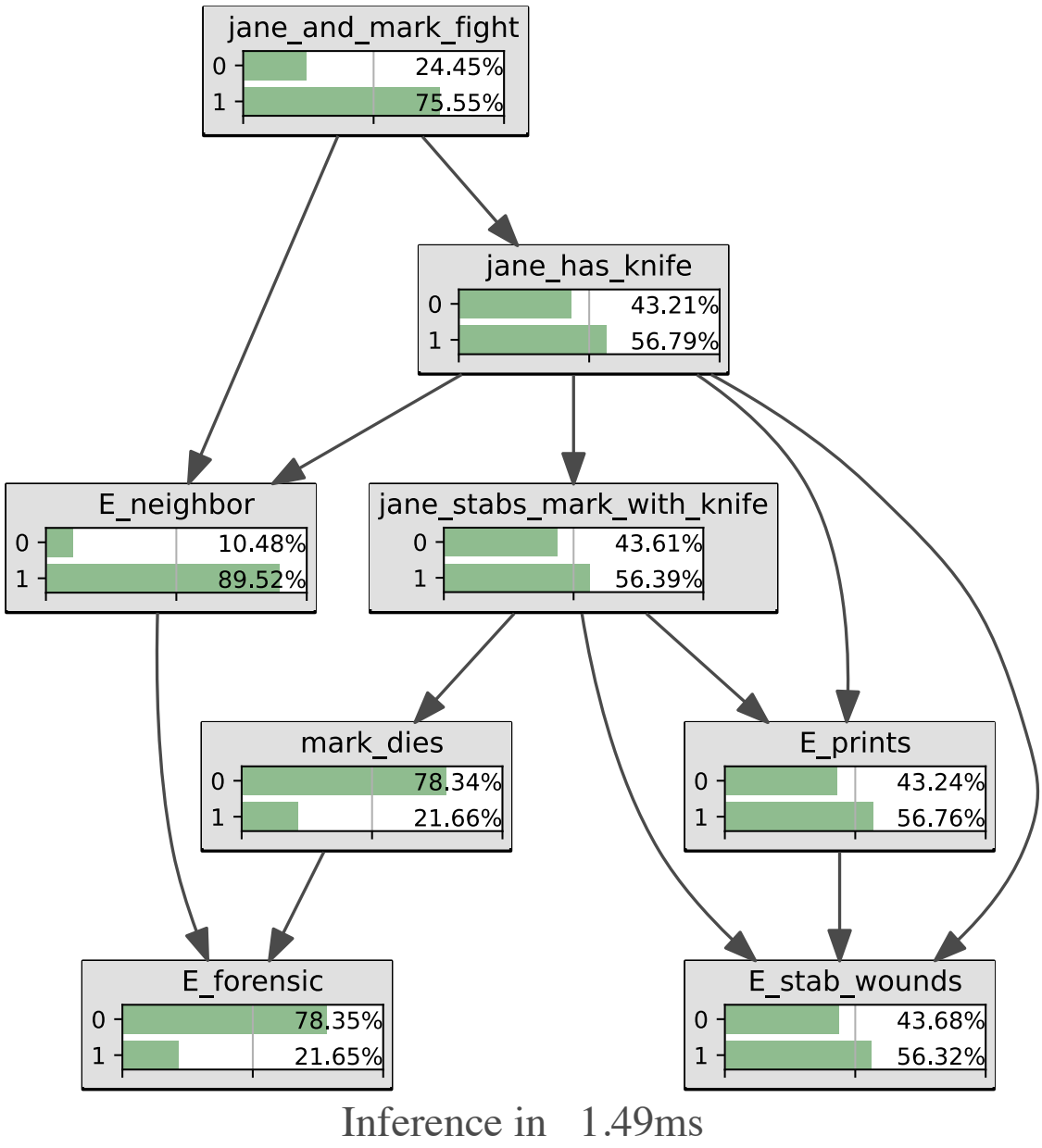

Figure 3.1: Walkthrough network.

3.5 Evaluating the Bayesian Network

Now we know that we can build Bayesian Networks automatically using the K2 algorithm based on data we generated in our simulation, we need to evaluate different aspects of the Bayesian Network.

3.5.1 Numbers

1. **Do the conditional frequencies in the BN correspond to the conditional frequencies in the simulation?**
2. **Could you find the numbers?**
3. **How robust are the BNs to reduced precision in the CPTs?**

3.5.2 Structural

1. **Does the BN represent all relevant events of the scenario?**
2. **Does the BN have temporal ordering of the hypothesis nodes?**
3. **Does the BN follow the evidence-idioms (evidence not parents of hypothesis nodes)?**
4. **Can the BN represent multiple alternative scenarios?**

3.5.3 Predictive

1. **How does the BN respond to evidence?**
2. **When does the BN predict the wrong outcome?**

Chapter 4

Grote Markt

4.1 Introduction

4.2 Evaluation

4.2.1 Numbers

1. **Do the conditional frequencies in the BN correspond to the conditional frequencies in the simulation?**
2. **Could you find the numbers?**
3. **How robust are the BNs to reduced precision in the CPTs?**

4.2.2 Structural

1. **Does the BN represent all relevant events of the scenario?**
2. **Does the BN have temporal ordering of the hypothesis nodes?**
3. **Does the BN follow the evidence-idioms (evidence not parents of hypothesis nodes)?**
4. **Can the BN represent multiple alternative scenarios?**

4.2.3 Predictive Skills

1. **How does the BN respond to evidence?**
2. **When does the BN predict the wrong outcome?**

4.3 Discussion

4.3.1 Random

1. **Does the BN not depend on private knowledge?**
2. **Could we apply this BN to a different location?**

Chapter 5

Conclusion

Our research questions, and answers:

- **Can we create Bayesian Networks that correctly reflect multi-agent simulations of crimes?**

  We can create Bayesian Networks that correctly reflect multi-agent simulations of crimes. For different types of simulations, the Bayesian Networks that were generated using the pipeline had high accuracy, low RMSE, generally fulfilled the structural criteria we set out. The probabilities in these networks reflected the probabilities we found in the simulation.

  The networks were even robust under imprecision, which means that even though experts might disagree on elicited probability values, or there are imprecisions in the measuring tools, the performance of the network and the conclusion that it gets to in the end, might remain correct.

  We can also show how we should update our posterior probability of an event given a set of evidence and hence show that different pieces of evidence have different strengths. The effect of evidence on the posterior lines up with the evidence as reported in the simulation, as well as with modeller's intuition about the evidence. The visualisation of the posterior probability of the outcome node is a useful interpretative tool for reasoning. Different sets of evidence can also be compared with each other, so that we could compare the effect of different evidence and different scenarios on the posterior probability of the ultimate output node.

  However, the networks that we generated using the pipelines did have some features that are hard on modellers, some nodes had too many parents to easily fill a cpt. We might want to trade-off high network scores for user ease, and restrict the number of

parents that a node can have. Additionally, the probabilities that were used in the network would be difficult for a human modeller to elicit or discover. However, since we have our robustness to imprecision, this might not be a large practical problem. Additionally, the temporal ordering of the nodes is not correct when two alternative scenarios are represented within one network.

- **If we can, under what assumptions do these networks function?**

  The assumptions under which the networks are correct, are too strict for this approach to be applied one-on-one to the real world. There are three problems that were identified in this work. These are the meaning of the reporters, private knowledge, and implicit conditioning. These problems are all abundantly present in the real world and it is not clear how to we should fix them. This means that Bayesian Networks might remain a good tool for seeing how evidence affects hypotheses, but might not be ready for the real world.

Reporters and random variables

Fundamentally, the problem is that the nodes in Bayesian Networks are random variables. They are not pragmatic, dialogical, argumentation, or logical sentences, but mathematical objects. They are random variables, and a random variable implies an observation procedure that maps a world state to a truth value. This means, that a node implies that we know how to measure if it is true or not in the real world.

In our simulation, this is really not a problem. We have an observation procedure: if a certain state occurs, we have the reporter in the same place as when the state change occurs, and the reporter reports exactly and only that. Hence, Bayesian Networks might work for subsections of reality that have a clear observation procedure. For example, reasoning with DNA evidence or other valid forensic evidence. In these cases, we know exactly what it means for the node to be true, since we know the measurement associated with the random variable.

However, for many of the events that we encounter in our simulation, or in Bayesian Networks in the state of the art, we do not know how we are determining that the node is true or not. We do not know which operationalisation is used to determine the truth value, and if that operationalisation is correct or not. This information has to be included with the networks, otherwise we cannot interpret what a node actually means.

Private knowledge

This problem is very clear: in our simulation, we know when private information is true or false, even if we have no way of knowing this private knowledge in real life.

If we remove this knowledge from our network, we find that this effects the posterior probability of the outcome node to such an extent that we fall below a 'comfortable' threshold of guilt. This implies that we can only generate a network that correctly and legally reasons about evidence if everyone in the process would help. This is not feasible.

Conditioning on implicit parameters

We find that if we change the parameters of the simulations, such as using a different map, the probabilities for events change. It means that there is outside influence - things that are not nodes, can influence the cpt's of the network, as if there is an invisible 'environment' node that is the parent of all the nodes in the network. This means that if we see a Bayesian Network and we do not know exactly the context in which it was created, we cannot assume that it generalises to a different environment, even if on the face of it, the nodes would be the same. We cannot take parts of that network (with probabilities) and put it in a different one, because the Bayesian Network is implicitly defined over the environment for which it was originally created, and not any other.

5.1 Future Work

It is difficult to assess the human criteria for Bayesian Networks. We need formal methods to investigate whether a person could really 'think' of a dependence relationship between two nodes, or estimate the correct probabilities. The ad-hoc, subjective way that the human criteria were assessed in this thesis are not very satisfactory. New standards need to be invented that correspond to objective facts about people's ability to estimate probability and investigate conditional relations between nodes.

The simulations as tested here are still very simplified - they are only appropriate as a baseline for testing Bayesian Networks, and for nothing else. Testing Bayesian Networks on more complex simulations would likely result in a reduced accuracy due to inherent and irreducible uncertainty, and lay bare many more problems that do not emerge from these simple simulations. Hence, testing on more complex models is necessary to fully understand the limitations of Bayesian Networks.

The values of ϵ can be reduced from 0.01 to smaller probabilities, to discover how this would influence accuracy and precision of networks, especially in situations that are more complex than the simulations we used here. If there are inherent features to the simulation that makes the frequency of some event smaller than 0.01, it is important to know how that would interact with an ϵ of 0.01. Hence, this should be investigated further.

As the K2 algorithm used depends on a given node ordering, the node-ordering algorithm

used in this project resulted generally in successfully replicating the temporal structure of scenarios, however the merging of scenarios caused problems, and resulted in a missing temporal structure. This can be fixed relatively easily, by taking the average time-step at which an event occurs into account, instead of only the ordering relative to other events.

While we discussed the problem of operationalisation in the discussion sections of these chapters, this was not investigated directly. This makes it unclear how big the effect of inefficient, insufficient or invalid operationalisation on the structural, performance and human criteria of the network would be. This problem can be investigated in simulation in simple ways - for instance, by changing the accuracy of reporting, such that a reporter might sometimes report an event wrongly. Alternative operationalisations for the 'same' random variables or events should also be investigated. Investigating the effects of non-ideal operationalisation is essential for knowing if we can generalise our Bayesian Networks beyond the bounds of simulation.

Testing or generating Bayesian Network idioms from simulations. The dream is to get "plug and play" Bayesian Network idioms - preconnected structures, perhaps even with some probabilities attached that you can add evidence to and adapt and combine if necessary. Using simulations, we can test the granularity of these possible idioms, to simulate a crime at larger and smaller resolution (more or fewer events) to see how well the idioms can capture it.