Evaluating Bayesian Networks as a Tool for Reasoning with Evidence using Agent-Based Simulations

Ludi van Leeuwen

Abstract

The existing methods for evaluating Bayesian Networks on full court cases are not sufficient, since the subjective probability assigned to the nodes might be overfitted on the world in which all pieces of evidence are true. This would result in incorrect outcomes for BNs where the state of the evidence turns out to be different. A systematic way of evaluating the response of a BN on all possible valuations of cumulative evidence is proposed in this paper, and illustrated by means of a simple agent-based simulation of a robbery. This method works correctly in the simulation, but is implausible to be generalisable to the real world due to combinatorial explosion of possible evidence states, the subjectivity in establishing a ground truth without using simulations, and the imprecision in node definitions in the BN in the first place.

Contents

1 Introduction

When we find evidence for a hypothesis that we have held in the back of our minds, our belief in that hypothesis increases. This is the essence of reasoning with evidence. Our goal in reasoning with evidence is to find a method that will lead us to believe as many true hypotheses as we can, given the evidence that we have.

However, the relationship between evidence and hypotheses is elusive. 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 can consider three main approaches for reasoning about hypotheses using evidence within the domain of AI and Law (Di Bello and Verheij, 2018): an argumentative, a scenario-based, and a probabilistic approach. One subfield of the probabilistic approach is the use of Bayesian Networks. Bayesian Networks can make explicit how our beliefs about events should change given that we find certain pieces of evidence for them. This idea of correctly updating on new information make BNs an interesting tool in the courtroom. A Bayesian Network might make transparent 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.

However, creating and using BNs is far from straightforward for both the builder and the interpreter. From de Koeijer et al. (2020): 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". This subjectivity is pervasive throughout the Bayesian Network: the events selected, arcs drawn, and numbers elicited are subjective and seem empirically untestable in the data-poor environment of AI and law. These problems might be mitigated in part by evaluation criteria.

These evaluation criteria, however, do do not always sufficiently safeguard against an incorrect network, which is a network that predicts the 'wrong' outcome based on some set of evidence. A BN for crimes should be accurate across all possible evidence states, since, if we want to use BNs in court and not model court cases afterwards, we do not know at the start what the state of the evidence will be. However, in general, it is hard for modellers to rationally consider all possible evidence states. A network might respond well to situations where all evidence as presented in the trial is present, but predict the wrong outcome when one evidence node is turned to a different value. This brings up questions about the use of BNs in general: do we really need to consider these states where the evidence is different from the evidence as presented in the trial? Since specifying the joint probability distribution is necessary in BNs, we have to specify all of these numbers. Anyway, if we want to take BNs seriously, they need to be correct over all possible evidence states.

The aim of this project is to define a method for the modeller to evaluate the BN over all possible evidence states, by grounding the BN in an agent-simulation of a crime. We can model crime cases using simulations to a level of realism we desire, measure the frequencies of events that emerge out of this simulation by running the simulation multiple times. In the agent-based system, we have full control over and full knowledge of the world that we are reasoning about, hence we can create a grounding that can be used to evaluate the BN. We can test whether our method of evaluating all possible evidence states works by comparing the predicted output of the BN with the frequency information about the output from the simulation. Then, we will discuss whether it is plausible that this method of evaluation generalises to real life[1].

something more general here about using simulations to test for BNS

[1]The simulation can be interacted with at `https://shielded-journey-34533.herokuapp.com` Code for this project can be found at `https://github.com/aludi/simulationTest`

2 Background

We want to create grounded Bayesian Networks for reasoning with evidence. This can be done by creating a agent-based simulation, building BNs based on the events in this simulation, and testing the evaluation method on that BN. In this section, the relevant background is laid out on reasoning with evidence, Bayesian Networks, existing evaluation methods for BNs, and agent-based simulations.

2.1 Reasoning with evidence

We consider three main directions in the field of reasoning with evidence within law (Verheij et al., 2015), (Di Bello and Verheij, 2018).

The first direction is via argumentation approaches, where hypotheses and evidence are represented as propositions that attack or support each other, as originating in (Wigmore, 1931) and in part unified by Dung (1995), for a historical overview see (Bench-Capon, 2019). An example of argumentation research in AI and law is ASPIC+ (Prakken et al., 2013).

The second direction is via scenario approaches, where coherent hypotheses are combined into stories (Pennington and Hastie, 1993), (Wagenaar et al., 1993), which are supported with evidence. Different types of stories need different pieces of evidence to be considered 'valid' or 'anchored' stories, and the extent to which different stories are anchored in the evidence determines how we assess them.

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, either by applying Bayes' law directly (Dahlman, 2020), or by using Bayesian Networks. Bayesian Networks can represent aspects of a criminal case or can attempt a scenario-like hybrid and represent the entire case. Examples of this are: modelling actual crime cases (Kadane and Schum, 1996), (Fenton et al., 2019), cases from fiction (Fenton et al., 2012) or fictionalised crime cases (van Leeuwen, 2019). The BN can also represent specific (forensic) aspects of a case, like DNA or blood-spatter evidence, for methods see (Meester and Slooten, 2021). Bayesian Networks can also integrate aspects of argumentation (Wieten et al., 2019), (Timmer, 2016), or of scenario-based construction (Vlek, 2016).

2.2 Bayesian Networks

A Bayesian Network (BN) is a directed acyclic graph that represents the joint probability distribution over a set of events (Pearl, 1988). BNs consists of nodes, arcs and conditional probability tables. A node is a random variable that represents an events. An arc represent a conditional relationships between two events, linking two nodes together. The two nodes are the parent and the child node, the probabilities in the child node are conditioned on the value that the parent node takes. The specification of these conditional probabilities is done in the conditional probability tables (CPT) of each node.

under construction

The combination of all nodes in the BN results in a full joint probability distribution over all nodes, which is

$$P(x_1, ..., x_n)$$

We can decompose this using the product rule

$$P(x_1, ..., x_n) = P(x_n|x_{n-1}, ...x_1)P(x_{n-1}, ...x_1)$$

Now, we see that $P(x_{n-1}, ...x_1)$ is also a joint distribution over all nodes, except node x_n. Hence, we apply the product rule again:

$$P(x_1, ..., x_n) = P(x_n|x_{n-1}, ...x_1)P(x_{n-1}|x_{n-2}...x_1)P(x_{n-2}, ...x_1)$$

Keep applying this until we're at

$$P(x_1, ..., x_n) = P(x_n|x_{n-1}, ...x_1)P(x_{n-1}|x_{n-2}...x_1)...P(x_2|x_1)P(x_1)$$

which is equivalent to

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|x_{i-1}, ..., x_1)$$

In BNs, we know by definition that

$$P(x_i|x_{i-1}, ..., x_1) = P(x_i|parents(x_i))$$

Hence,

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|parents(x_i))$$

For every node x, hence, we will fill out the CPT

$$P(x_i|parents(x_i))$$

We show an example BN in Figure 1. This is the basic form of the evidence idiom (Fenton et al., 2012): we have a cause as a parent node, and a consequence as a child node. In this context, we want to interpret the parent node as a 'hypothesis', that we want to prove, but cannot observe directly. The child node is a piece of evidence that we can observe: it is either true, or false. Specifically instantiated, like in Figure 1, we create a basic scenario: 'suspect shot victim with gun' as the parent (hypothesis) node, and 'bullet casings found on ground' as the child (evidence) node. We can use this network, with the probabilities as defined in the CPTs, to reason about the hypothesis given the evidence that we find.

In Figure 1, the node 'suspect shot victim with gun' is a hypothesis. Nodes with one or more incoming arcs are conditionally dependent on their parent node(s). In our example network, the node 'bullet casings found on the ground' 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 (Dawid, 2008).

The arcs and the CPTs tell us how we should reason about the hypothesis, given that we find some piece of evidence.

For a simple example, for a node with only one parent, we can just use Bayes law directly.

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

To update to our network let us say that we found B, so there are bullet casings on the ground. We want to know what this new evidence will do to our belief in S. This means that we are updating our belief in S, or finding the posterior of S, based on the evidence that we found B.

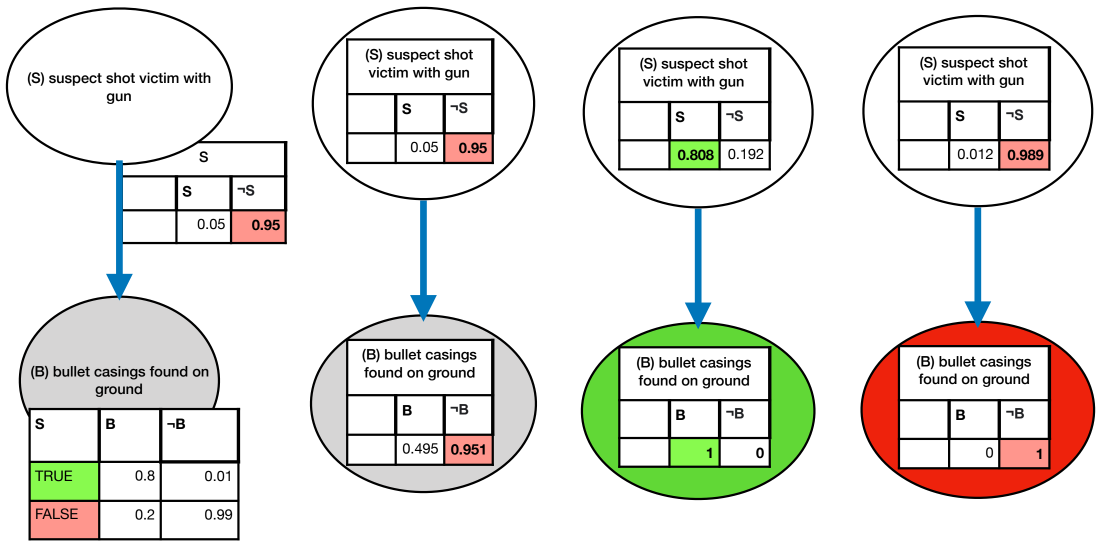

(S) suspect shot victim with gun
S
S
¬S
0.05
0.95
(B) bullet casings found on ground
S
B
¬B
TRUE
0.8
0.01
FALSE
0.2
0.99
(S) suspect shot victim with gun
S
¬S
0.05
0.95
(B) bullet casings found on ground
B
¬B
0.495
0.951
(S) suspect shot victim with gun
S
¬S
0.808
0.192
(B) bullet casings found on ground
B
¬B
1
0
(S) suspect shot victim with gun
S
¬S
0.012
0.989
(B) bullet casings found on ground
B
¬B
0
1

Figure 1: Inference with Bayesian Networks: Defining the network with CPTS, entering no evidence, entering positive evidence, and entering negative evidence

$$P(S|B) = \frac{P(B|S) \cdot P(S)}{P(B)}$$
$$P(B|S) = 0.8, P(S) = 0.05, P(B) = 0.8 \cdot 0.05 + 0.01 \cdot 0.95$$
$$P(S|B) = \frac{0.8 \cdot 0.05}{0.8 \cdot 0.05 + 0.01 \cdot 0.95} = 0.808$$

So now we have found the posterior of S, our belief in S has increased from its prior probability of 0.05, to its posterior probability of 0.808, given evidence B. On the other hand, if we do not find bullet casings on the ground, our belief that the suspect shot the victim decreases from 00.5 to 0.012.

Hence, we can exactly specify how our belief in a hypothetical event changes given all possible values of the evidence.This is the simplest form of Bayesian reasoning, with one piece of evidence, and one hypothesis. The general form of the update is more complicated (see ....).

In real-life situations, we are constantly reasoning with many pieces of evidence that might be conditioned on more than one parent node, resulting in tedious and error-prone calculations if we would do them manually. This is where Bayesian Networks are useful.

We can outsource the Bayesian calculation to the BN, however, the network itself has to be constructed in some way. This can be done by hand by modellers, in (proprietary) software with a GUI like AgenaRisk or Hugin. They can also be built, by hand, or automatically constructed from datasets in PyAgrum, (Ducamp et al., 2020), a free Python software package. In this project, PyAgrum was used.

2.2.1 Evaluating Bayesian Networks

Bayesian Networks have been used to model reasoning with evidence in criminal cases, although these methods have not been used in court. However, even though there are methods for building BNs, it is unclear how we should validate and evaluate them.

In data-rich domains, we can train Bayesian Networks on large amounts of frequency data and estimate the accuracy of the BN by cross-validation. We define a training set and a test set of the data (usually a 80/20 split). We construct the network based on the training data, 80% of the input. Then, we test if the network learned the correct conditional probability relations on the final 20% of the data, by comparing the expected output in the test data to the predicted output from the BN (Hamilton and Pollino, 2012). The 80/20 split allows us to measure to what extent the network is performing well on the entire dataset, and not just overfitting on the data that it has access to.

However, this approach towards BN validity through accuracy is implausible in BNs for court cases. The court cases involve one-of-a-kind events (Schum and Martin, 1982), which means that it is unlikely that we can find frequency data on most of the nodes present in our BNs. This means that we cannot do an 80/20 type split in our data, as we only have one piece of data, which is the court case that we are modelling. This is why the probabilities in the BNs for court cases not pure frequencies, but instead subjective probabilities based on estimated frequencies, either elicited from experts or estimated by the modeller.

There are other ways of evaluating BNs than accuracy. The evaluation methods used by Fenton et al. (2019) and Vlek (2016), who both model BNs based on real case studies, confine themselves to two approaches:

1. **Sensitivity analysis**

   Sensitivity analysis tests the effect of small parameter changes in the CPTs of nodes on the outcome of the network. This is useful if the outcome of the network hinges on elicited or subjectively estimated probabilities. A sensitivity value of near 0 means that at the point of the CPT, a ‘small’ change in the parameter would result in an infinitesimally small change in the output probability. That the CPT is not sensitive at all and a misestimation might not matter so much. However, a large sensitivity value means that the value is very sensitive to change : a small change in CPT might result in a meaningfully different output. Since Bayesian Networks have many parameters per node, it is not trivial to perform a sensitivity analysis that takes into account all the nodes in the network, and interpret the results in a meaningful way - this would be an n-way sensitivity analysis (Van Der Gaag et al., 2007).

   In Vlek (2016), sensitivity analysis is mentioned, but not applied. On the other hand, in Fenton et al. (2019), each individual hypothesis node is subjected to sensitivity analysis, to find out for how each of the parameters in the nodes need to change in order to result in a 0.95 or a 0.99 posterior for guilt. There are only a few ways that individual nodes can change that would result in a high probability of guilt, from which the authors interpret that the constructed BN is robust to small inaccuracies in subjectively elicited CPTs. However, in the paper, this is only shown for one evidence state. It is not clear how the BN would respond to ‘counterfactual’ pieces of evidence.

2. **Cumulative evidence**

   Another way of evaluating the response of the BN, is by seeing its response to cumulatively setting evidence. This way, we can see how much each piece of evidence (from the defence or the prosecution) changes our belief in the posterior of the output node(s). We could subjectively assess if it makes sense that a certain piece of evidence has a large, or small, effect on the posterior. In both (Fenton et al., 2019) and (Vlek, 2016) this cumulative approach shows that, in general, the evidence from the defence decreases our belief that the suspect is guilty, and evidence from the prosecution increases our belief that the suspect is guilty. However, in both cases, the BN is only evaluated on one state of evidence. Both cases only test the cumulative probability of the output node by turning the evidence nodes to values that correspond to the findings of the court. However, this is a risky move, since this might lead to overfitting the probabilities in the CPTs, so that the network seems to reflect the right ‘evidence strength’ on the court-found state of evidence, but is actually misrepresenting other combinations of evidence, without a way of checking if this is the case.

Our findings in Figure 11 show that it is necessary to consider all possible combinations of evidence. If we have tunnel vision and only focus on the state of evidence that is found in court, we might create a network that overfits on one state of evidence. This means that if there was a re-trial and the court would decide that some pieces of evidence are not admissible after all, the BNs might show the unreasonable behaviour. Additionally, if we are building the BN before or during the court case, we cannot know yet which state we are in, as none of the evidence has been established yet. Therefore, the network should work correctly over all possible evidence states. If we want to take the premise behind BNs seriously, we have to create a network that gives sensible results for all possible combinations of evidence, not just the valuation of the evidence that is initially established in court.

2.3 Simulation

We are using agent-based simulations to investigate methods for evaluating Bayesian Networks. Agent-based modelling is a modelling primitive that allows researchers to study models in which agents interact with their environment and with other agents (Gilbert and Terna, 2000).

Agent-based modelling is useful for simulating criminal behaviour in spatially explicit ways. The actions of a thief and victim are constrained by geography and local knowledge. For instance, a thief cannot move through buildings like a ghost, or steal from someone they have not seen. Due to specific interactions that arise out of the circumstances in each run of the simulation, we can collect frequency information about all the relevant events in our modelled scenario that would be hard to estimate with subjective probabilities or with mathematical (non-spatially explicit) models. Since we have full knowledge of the frequency of events in the simulation, we can use it as a 'grounding', or baseline. In a sense, the simulation is a data-generating environment for criminal cases. If we look at the taxonomy of agent-based model levels in (Gilbert and Troitzsch, 2005), where level 3 is a simulation that really reflects the real world, and level 0 is a 'caricature of reality, as established with simple graphical devices', the simulation in this paper is at level 0, to reduce unnecessary modelling and time complexity. In this project, the simulations are programmed in Python using the MESA framework (Kazil et al., 2020).

3 Method

Our goal is to establish a method for evaluating BNs by preventing overfitting on one evidence state. We can consider an approach in four stages.

1. *Scenarios* We need a 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 prosecution and defence.

2. *Simulation* Based on this written description of the scenario(s) to be modelled, an agent-based simulation is created, by first identifying and operationalising all relevant agents, objects and environments, and secondly identifying and operationalising all relevant events. Depending on the goal and complexity of the scenarios to be modelled, the granularity and overall level of the simulation should be considered. The simulation should be repeated until we find the limiting frequency of all events. These frequencies are collected.

3. *Construction* The collected frequencies are used by the K2 algorithm to automatically build a Bayesian Network of the simulation.

4. *Evaluation* The generated Bayesian Network is evaluated on all possible states of the evidence.

3.1 Scenarios

We start our process with one or more written scenarios. These scenarios can be obtained from abridged or simplified court case descriptions of the crime, or they can be wholly fictional (such as in this paper). The scenarios should contain all and only those hypotheses and evidence that are relevant to the case. Both the prosecution and the defence should be able to select relevant events and their evidence.

3.2 Simulation

We can think of the simulation as having two parts: a model of reality, and a way of observing the model by observing variables that the modellers deem interesting. In this paper, the model of reality is an agent-based simulation, and the observation procedure are random variables (or, reporters), which become nodes in the BN.

The agent-based simulation is a simulation that should include the possibilities for all the scenarios as outlined in step 1. It should contain agents, objects and an environment. The level of detail or real-world validation at which agents, objects and environments are modelled depend on the purposes of the environment.

go into more detail about constructing simulation here, use the BG

In the simulation, certain events can be brought about. Any one of the above-mentioned events can happen. 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). In short, a random variable is a function that maps an event (e) to a truth value:

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

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:

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

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:

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

3.3 Construction

The output of this method is the collection of runs O, where each run is the global state G of the simulation, as measured by the random variables R. These random variables 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.[2] The algorithm used in this experiment to structure the Bayesian Network from the simulation data is the K2 algorithm (Cooper and Herskovits, 1992).

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 (Chen et al., 2008).

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. By adding the nodes to the simulation in the relevant order, we can ensure that the order is temporal.

3.4 Evaluation

We can evaluate the response of the Bayesian Network using the effect of cumulative evidence on the posterior probability of the output nodes. This means that we order the evidence nodes chronologically, then instantiate every evidence node to a truth value (either T or F) in order. Then, we measure the posterior probability of all output nodes. This results in a diagram or table where we can see the evidence strength of each piece of true or false evidence reflected. We propose a method for evaluating Bayesian Networks over all possible global evidence states in this paper. This method is described here.

1. Select all evidence nodes (a total of e states).
2. Order the evidence nodes chronologically.
3. Create a baseline 1) Select all possible evidence states. An evidence state is a valuation of evidence, where for every evidence node, a truth value is specified. For this simulation, an evidence state could look like (1, 0, 0, 1), which would correspond to finding that the psychological report is true, the agent was seen

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

on camera, but not seen stealing, and the object was lost.

If there is enough data available, as is the case in our simulation, we can extract all possible evidence states automatically. If we do not have enough data, we need to manually consider each evidence state (start with 2^e states, ask an elicitor to delete states that are not possible).

4. Create a baseline 2). For each possible evidence state, create a total probability distribution over all possible output states. Speaking generically about two alternative, non-mutually exclusive outcomes scenario 1 (scn1) and scenario 2 (scn2), we need to find a probability distribution over $scn1, \neg scn1 \wedge scn2, \neg scn1 \wedge \neg scn2$ that is constrained by $P(scn1) + P(\neg scn1 \wedge scn2) + P(\neg scn1 \wedge \neg scn2) = 1$. Or, a weaker version, we create a preference ordering on $scn1, \neg scn1 \wedge scn2, \neg scn1 \wedge \neg scn2$, without specifying probabilities.

   In our simulation, or in data-rich environments, we can automatically create this total probability distribution, since we can count the conditional frequencies that occur in simulation. In real life, a preference ordering or probability distribution over the possible outcomes would need to be elicited by modellers from experts.

5. For every possible evidence state, turn the relevant nodes in the BN on in order. At each step, record the posterior probability over the output nodes. Automatically convert the probability distribution over the outcomes into a preference ordering.

6. Compare the BN prediction of output nodes given evidence to the prediction establishes in step 4). Once all evidence is added, we can calculate the divergence of the BN prediction and the actual frequency in the simulation by

   $$|P(output|E) - F(output|E)|$$

   , where E is the total set of evidence in a given state.

7. Calculate the total accuracy per output node by

   $$\frac{\sum_{i=0}^{e} |P(output|E_i) - F(output|E_i)|}{e}$$

   For the preference ordering, we do the same, only we calculate a counting score: when the preference ordering extracted from the BN is the same as the one from the simulation, we give a point, otherwise we do not, then average this over all sets of evidence.

4 Case Study: Robbery at Grote Markt

We illustrate the method by means of applying it to a case study of a robbery at the Grote Markt, the central square in Groningen.

4.1 Experimental Set-up

BNs were created and evaluated based on different number of runs, to evaluate how the accuracy of the BN depends on the data available to it.

We want to create both a ground truth, that are the frequencies that we're interested in, and then the probabilities that are going to approximate it so that we can check the accuracy of the network. To find the ground truth, we need to find the limiting frequencies of the events in the simulation, the simulation was run 10,000 times. One thing we want to find out is the necessity of precision on the outcome of the networks. This was done by changing the number of runs that the BN was created on. The different number of runs of the simulation were [1, 5, 10, 20, 40, 60, 100, 200, 500, 1000] for creating the BNs.

Every simulation was run for 100 steps, or until both agents were in their target states.

4.2 Scenarios

We have established three different scenarios. In all scenarios, there are two people walking around the Grote Markt. One person is young, and the other person is old and carries a valuable object.

In scenario 1 (scn1), the young person sees the old person, assesses whether the object is valuable enough to risk stealing, and if the old person is vulnerable enough to steal from, and a motive is established. Then, if the young person has a motive, they will attempt to sneak up on the old person. When they get near, they steal from the old person.

In scenario 2 (scn2), the young person might still be doing all of the above (or they might not), however, before they can steal, or if they decide that they're not stealing at all, the old person drops the object accidentally.

In scenario 3, neither scenario 1 or scenario 2 happens. The people walk around at Grote Markt and then go home. We do not have to represent scenario 3 explicitly, it can be represented as neither scn1 nor scn2.

For our evidence, we have a psychological assessment of the young person, that assesses whether they are psychologically capable of stealing from the old person. We also have cameras at Grote Markt that show whether the young person was seen at all, or was seen stealing. Additionally, we have the fact that the object is gone or not.

4.3 Simulation

Now we need a way to transform the natural language descriptions of the events in the scenarios to observable events in the simulation. First, we identify all relevant actors, objects and environments in the scenario and operationalise them in a simulation. Second, we identify the relevant reporters.

4.3.1 Agents

Agents can interact with other agents and with their environment. Every agent has features, functions and roles associated with them. The agents in this simulation are created from the MESA agent class. These are goal-based agents. Based on the scenarios as described in the previous step, we want to create one older, vulnerable agent, and one younger potential thief. In principle, the simulation is flexible enough that any agent can be a thief, as motive is determined by a risk-trade-off and not something inherent to the agents themselves. However, since we wanted to avoid problems with identity and follow the scenarios, we made sure that only one agent would be the thief, by having it have an object that was not valuable and an age that was not vulnerable. The behaviour of the agents is depicted in Figure 2.

All agents have the following **attributes**:

role Every agent has a role of either victim or thief. In a more realistic scenario this role would be assigned based on behaviour, but in this simulation, this was done at the start (because the scenario allows for one thief, and one victim). However, the role 'thief' does not mean that the agent will actually steal, and the role 'victim' does not mean that the agent will be robbed[3]: if the victim drops the object, or if the thief and victim never meet, these roles are not fulfilled by the agents. Hence, the role only implies a certain potentiality.

id Every agent has an ID. The thief has ID 1, the victim has ID 0.

[3]From now on the '' around the roles are dropped

object The thief has an object with a value of 0, the victim's object has a value drawn from a uniform distribution between 500 and 1000.

goal location The goal location of the agent is either it's initial goal location that was assigned at the start, or, when the agent has a motive, it is the current location of the victim agent. The initial goal location for each agent is drawn randomly from any of the accessible locations at the edge of the map.

age The thief's age was 25, the victimt's age was drawn from a uniform distribution between 60 and 90.

risk threshold The risk threshold for the victim agent was set to 5000, such that it is never worth for it to steal, the risk threshold for the thief is randomly drawn from a uniform distribution between 800 and 1200.

age threshold The age threshold signifies at what age an agent considers another agent vulnerable: older agents are more vulnerable than younger agents. The victim's age threshold is set to 100, it will only steal from agents that are 100 years or older. The thief's threshold is randomly drawn from a uniform distribution between 50 and 100.

steal state The possible steal states are: MOTIVE, SNEAK, STEALING, SUCCESS, N, DONE, LOSER.

All agents have the following **actions**:

hang around Agents move around randomly.

walk Using standard mesa functionality, at every step of the agent, the agent moves 1 cell in the Moore neighborhood. The agent attempts to walk towards its goal state, whether that is a moving agent or their goal state. They move towards their goal state by taking the possible step that minimises the distance between their goal and themselves using the Euclidian distance as measure (this does not take buildings into account).

escape Given the use of Euclidian distance instead of a smarter, geographically-informed approach (using waypoints or something similar), agents can get trapped in tight corners of the simulation when they're moving towards their goal. There is an epoch tracker that counts 'stuckness' and then moves the agent into the 'hang around' state, of random movement, for a given time. This means eventually the agent will randomly move away from the 'stuckness'.

see every agent has a visual range of 10 cells. This means that all simulation objects (agents, objects) within that radius are selected. Actual vision is only calculated based on line-of-sight, which means that for each object in range, we use the Bresenham's line algorithm to select the relevant cell that lie on the 'environment grid' in between the agent and the object that it is seeing. For every cell on the straight path between the two objects, we check whether it is 'accessible' or 'non-accessible'. Agents, and light, cannot pass through 'non-accessible' cells. This is an abstraction, in real life there are windows. So if there's one 'inaccessible' cell on the list of cells that are the straight line between seeing agent and object, the agent is not able to see the object.

decide valuable Once the thief has a targeted victim, the thief knows the value of the object of the victim. In this function, the agent decides whether the object is worth stealing. If the value of the object is larger than the risk threshold of the agent, the object is deemed valuable.

decide vulnerable Once the thief has a targeted victim, the thief knows the age of the victim. In this function, the agent decides whether the victim is vulnerable enough to steal from: if the age of the victim is older than the age threshold, the thief considers the victim vulnerable.

sneak Agent moves to the position of the victim, only it can move with a radius of 2 (hence, at higher speed).

steal Whenever the thief is in the same cell as the victim - and they're both still within the simulation (so the victim has not reached its goal yet), the thief steals successfully.

4.3.2 Environment

The environment is a discrete grid of size $x = 75$, $y = 50$ that represents the geography of the Grote Markt. The real area of interest is approximately 425m x 280m, as estimated by the distance tool on Google Maps. This means that one cell in the simulation is equivalent to a square of 5.6m x 5.6m in real life. An agent can move 1 cell per time-step, which means that, given an average human walking speed of 1.4m/s [4], one time-step is equivalent to 4 seconds in real life. For the purposes of this simulation, this spatial and time resolution is high enough.

To simulate which parts of the Grote Markt were accessible to the agents, we had

[4] https://en.wikipedia.org/wiki/Preferred_walking_speed

to convert a map image of the Grote Markt into an agent-readable world. This was done by writing a method to convert screenshots of maps into an agent-readable environment. The map was screenshotted from http://maps.stamen.com/terrain/#18/53.21618/6.57225 and converted into greyscale. A grid was overlaid on the image. On the greyscale map, the color of the buildings was in the range of (189, 199): cells within this range were coded as 'inaccessible', since agents cannot walk through buildings. All other cells were 'accessible'. This resulted in a map shared by all agents that constrained their movements. The map constrains both vision and movement of the agent: we used this map to calculate the sight lines of both cameras and agents. An agent or a camera can only see another agent if there are no 'inaccessible' grid cell on the sight line between the two, and the other agent is within their visual range. The environment is shown in Figure 4.

4.3.3 Objects

There are two types of objects in the simulations, which are the valuable object, and the cameras.

In the scenario, an object is described as being in the possession of agents, or lost accidentally. In this simulation, the object was operationalized as being a feature of the agent, and not as an individual thing by itself.

There are 5 cameras in the simulation. They are placed randomly on the 'accessible' cells on the map. Every camera has a visual radius of 5, corresponding to a range of 28m, which is about equivalent to the visual range of real-life security cameras [5].

4.3.4 Reporters

We divide all all events into three classes: either an event is 'evidence' (E), a 'hypothesis' (H), or an ultimate 'outcome' (O).

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 Gerritsen (2015)), or with simple behavioural 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.

[5] https://securitycamcenter.com/how-far-can-security-cameras-see/

The operationalisation of the random variables is described below.

(H) motive_1_0 agent 1's steal state is MOTIVE.

(H) sneak_1_0 agent 1's steal state is SNEAK.

(O) stealing_1_0 agent 1's steal state is STEALING.

(O) object_dropped_accidentally_0 At every epoch, there is a 1/500 probability that the victim agent will drop the object by accident. If the object is dropped, this reporter is turned on.

(E) E_camera_1 agent 1 is seen on any one of the cameras.

(E) E_psych_report_1_0 if the thief has a motive, we draw an estimated risk threshold and an estimated age threshold from two normal distributions (mean = victim age, sd = 20), (mean = value good, sd = 100). If the victim is older than the thief's minimal age threshold, and the object more valuable, we estimate that the psych profile states that the victim fits the thief's profile, else not. This is not represented spatially in the simulation itself.

(E) E_camera_seen_stealing_1_0 if any camera sees agent 1 when agent 1's steal state is STEALING.

(E) E_object_gone_0 if the object is dropped accidentally, or if the object has been stolen.

A global state where the thief stole the object and all the evidence pointed in this direction, would be represented as (reporters in the same order as in the listing above):

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

4.4 Construction

The baseline was created by running the simulation for 10,000 runs. Then, to test to what extent the number of runs (=the amount of data available to the K2 algorithm) influences the accuracy of the network, for each of [1, 5, 10, 20, 40, 60, 100, 200, 500, 1000] runs, the collection of runs O was passed to the K2 algorithm, that generated a network.

The K2 algorithm requires an ordering on the nodes. In all number of runs, this ordering was [object dropped, motive, sneak, stealing, psych report, camera, cameraStealing, object gone]. This ordering was determined by considering the hypothesis-nodes

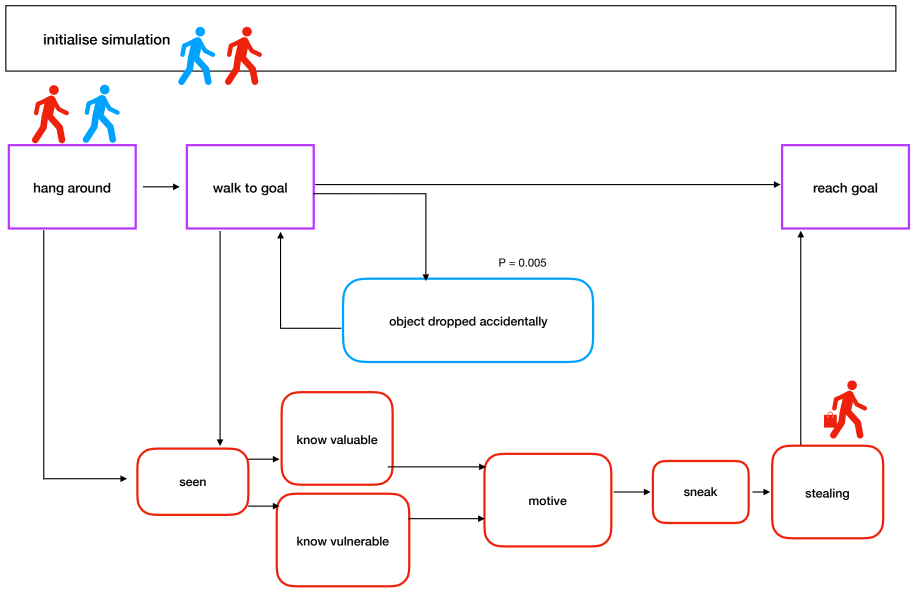


Figure 2: The behaviour of the agents. The nodes with rounded edges correspond to the nodes in the Bayesian Network.

There are two agents, a thief and a victim.

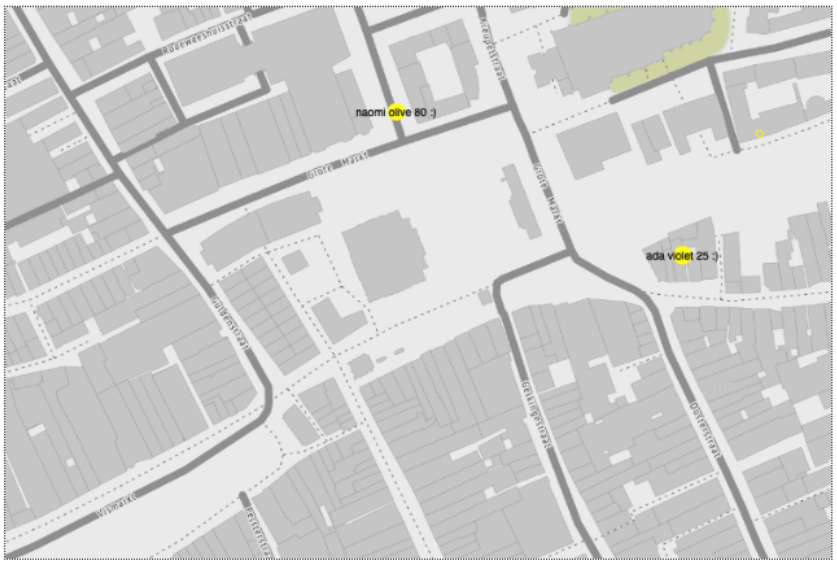


Figure 3: map of environment - 2 agents

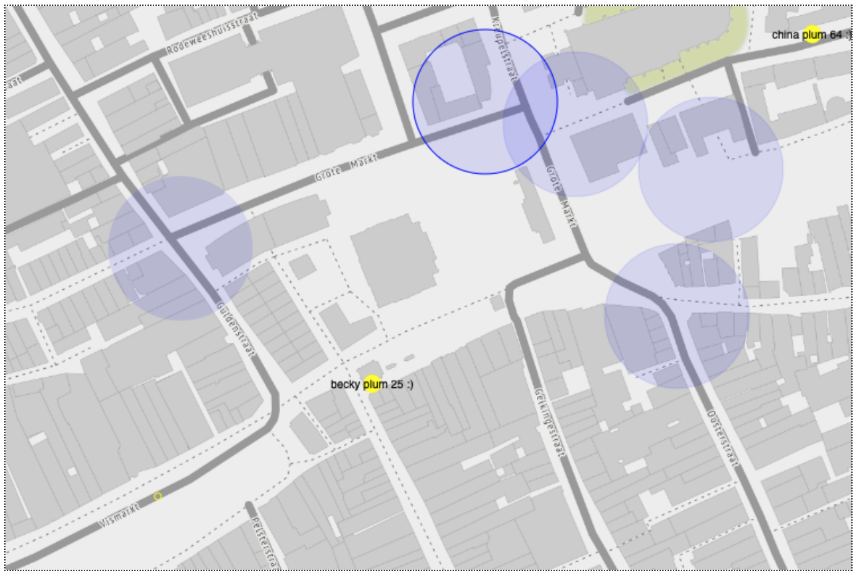


Figure 4: Camera locations are randomly initialized

first, and all evidence at the end. This is because evidence nodes are usually children of the hypothesis nodes. The hypothesis nodes belong to two different scenarios: scenario 1 is the motive-sneak-steal scenario, and scenario 2 is the object-dropped scenario. Placing object-dropped/scn 2 first, instead of 'motive'/scn1 is not arbitrary, there is a causality at play: if the agent drops the object, the thief cannot have a motive to steal the object, yet, having a motive does not 'cause' or 'condition' the object to drop. Hence, the 'accidental drop' could be the parent node (if we're taking a causal interpretation).

4.5 Evaluation

1. The evidence nodes of the network are the 4 (binary) nodes marked with an 'E', which are [E_camera_1, E_camera_seen_stealing_1_0, E_object_gone_0, E_psych_report_1_0].
2. The chronological order of this according to the simulation is: [E_camera_1, E_psych_report_1_0, E_camera_seen_stealing_1_0, E_object_gone_0,]
3. We create a baseline by running the simulation 10,000 times. We assume that events have reached their limited frequency at this point and that all possible states have occurred. In essence, there are $2^4 = 16$ possible states, but we can eliminate 7 of them, because they never occur in simulation.

   0000 ok 0001 ok 0010 no 0011 no 0100 ok 0101 ok 0110 no 0111 ok 1000 ok 1001 ok 1010 no 1011 no 1100 no 1101 ok 1110 no 1111 ok
4. We create the baseline by running the simulation. Baseline in Table 8.
5. For every evidence state, the nodes were turned to those valuations in update. We used the LazyPropagation (exacy inference) inference engine in Pyagrum.
6. Tables and averages in the Results section.

5 Results

TO BE UPDATED and EXPLAINED

We have a final Bayesian Network (Figure 5), that was created from the K2 algorithm that contains all the nodes specified. The arcs between nodes are not causal but conditional.

evidence	H1	H2	H3
(0, 0, 0, 1)	F(dropped) (0.99)	F(steal) (0.01)	F(neither) (0.00)
(0, 0, 0, 0)	F(neither) (1.00)	F(steal) (0.00)	F(dropped) (0.00)
(0, 1, 0, 1)	F(dropped) (0.99)	F(steal) (0.01)	F(neither) (0.00)
(1, 0, 0, 1)	F(steal) (0.93)	F(dropped) (0.07)	F(neither) (0.00)
(0, 1, 0, 0)	F(neither) (1.00)	F(steal) (0.00)	F(dropped) (0.00)
(1, 1, 0, 1)	F(steal) (0.93)	F(dropped) (0.07)	F(neither) (0.00)
(1, 1, 1, 1)	F(steal) (0.95)	F(dropped) (0.05)	F(neither) (0.00)
(1, 0, 0, 0)	F(neither) (1.00)	F(steal) (0.00)	F(dropped) (0.00)
(0, 1, 1, 1)	F(steal) (1.00)	F(dropped) (0.00)	F(neither) (0.00)

Table 1: Preference ordering with numbers for all evidence states on 'ground truth' for runs=10,000

We have passed the structure of the network to the K2 algorithm (not automatically).

This results in networks like this.

We have a plot of the overall accuracy over all evidence sets of the network for both the 'stealing_1_0' node and the 'object_dropped_accidentally_0' node (Figure 6). For a low number of runs, the network could not be created, hence the accuracy of 0. At around 10 runs, a network could be created, with an accuracy on all cumulative evidence states of around 0.75 for both nodes. Increasing the number of runs, hence, the data available to K2 to create the network, results in a higher accuracy. We have a frequency distribution over the possible states (Figure **??**), where we see that most possible states occur infrequently, while one possible state [0, 1, 0, 0] occurs 70% of the time.

In Figures **??**, **??**, we show cumulative evidence plots on both output nodes for two sets of evidence, on both a BN that was created from 10 runs, and on a BN that was created from 1000 runs of the simulation. Every plot shows the BN predicted output (bar on the left side), next to the actual frequencies in the simulation (bar on the right side).

In Tables 3, 4, we show the final cumulative probability and preference ordering for the output nodes as the worst (10 runs) and best (1000 runs) BN predicted them. In Table 8, we show the frequency of the output for each possible evidence state, as well as the preference ordering, for the ground truth as found in the simulation.

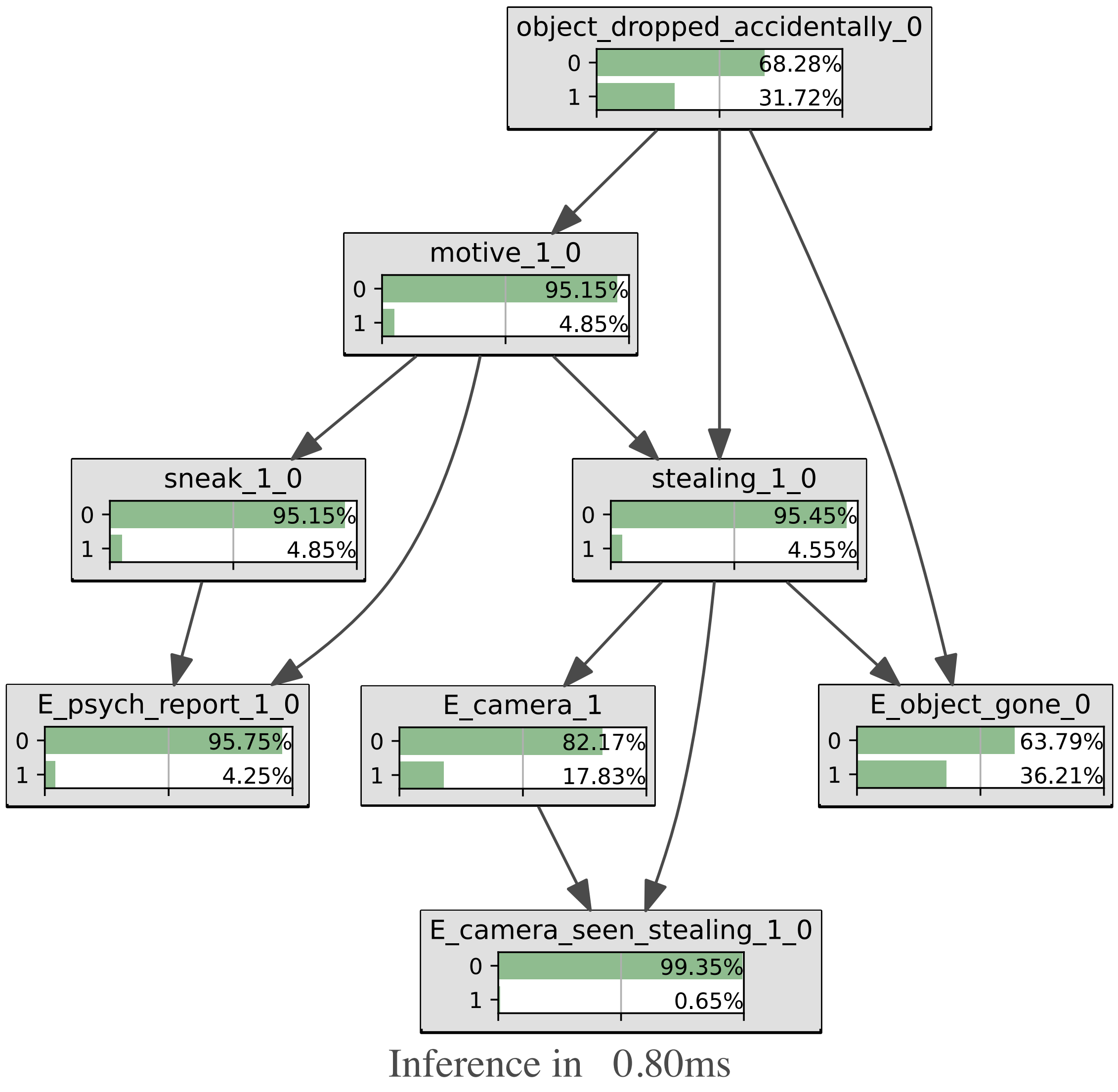


Figure 5: The final network (1000 runs)

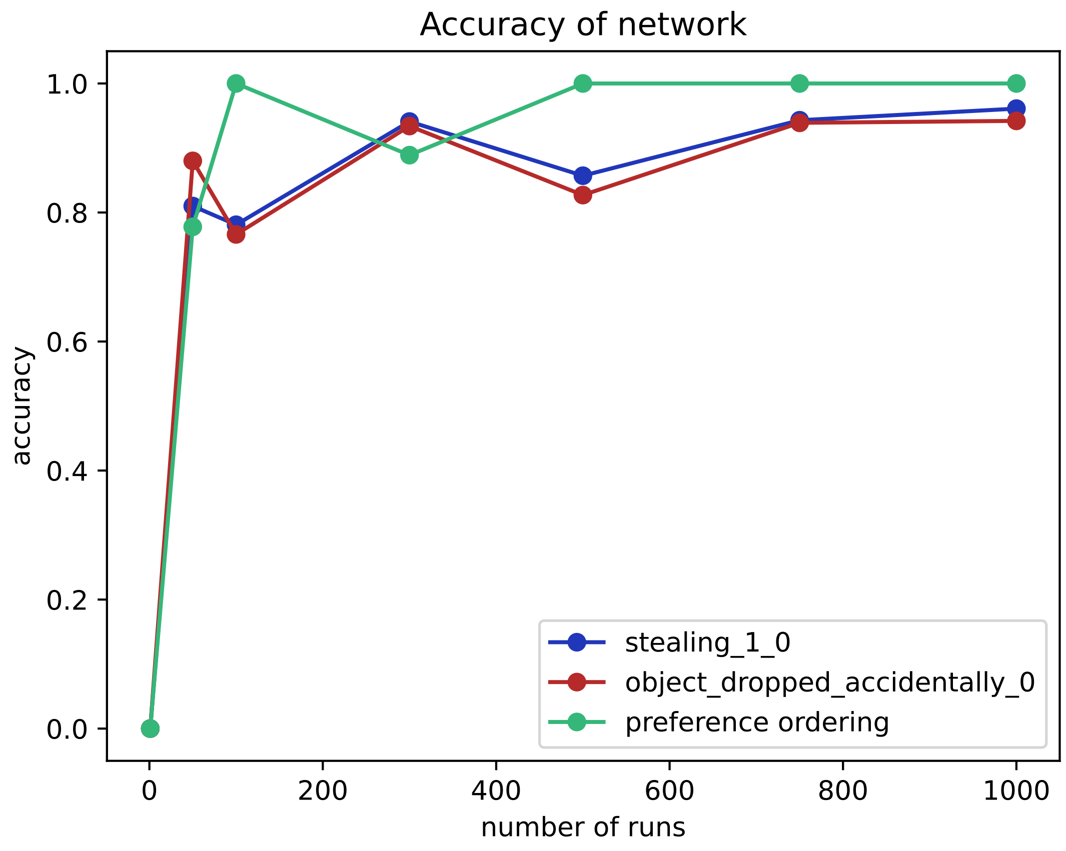
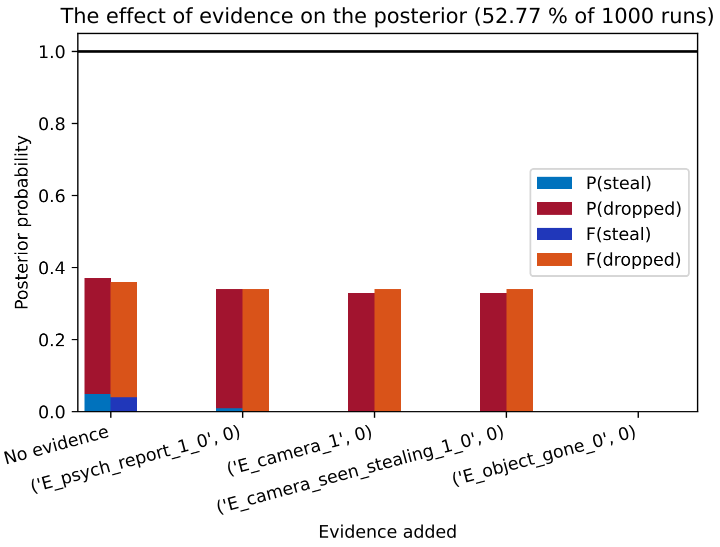
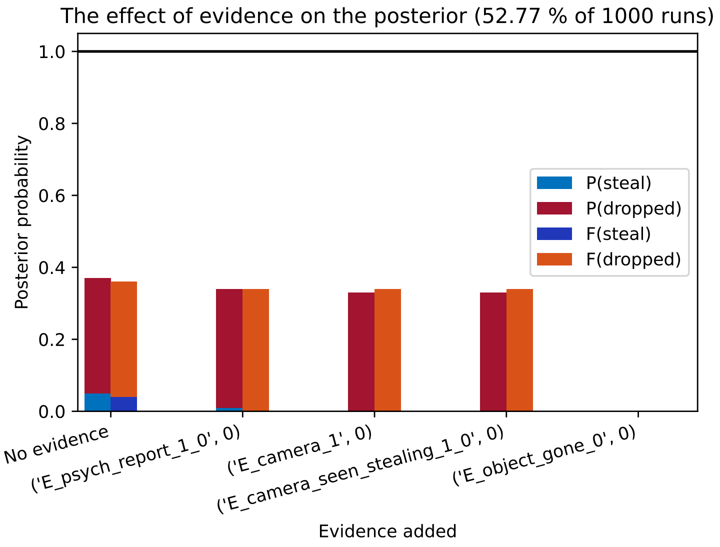


Figure 6: Accuracy of network based on the number of training runs



(a) Most relevant



(b) Most frequent

runs	50		100		300		500		750		1000	
evidence	S	D	S	D	S	D	S	D	S	D	S	D
(0, 0, 0, 1)	0.99	1.00	1.00	1.00	0.99	0.99	0.98	0.98	0.99	0.99	0.99	0.99
(0, 0, 0, 0)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(0, 1, 0, 1)	0.98	0.98	0.99	0.99	0.99	1.00	0.98	0.98	0.99	0.99	0.92	0.92
(1, 0, 0, 1)	0.47	0.21	0.66	0.64	0.98	0.97	0.95	0.95	0.94	0.94	0.96	0.96
(0, 1, 0, 0)	0.89	0.85	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(1, 1, 0, 1)	0.97	0.86	0.20	0.20	0.75	0.75	0.95	0.95	0.94	0.94	0.94	0.94
(1, 1, 1, 1)	0.96	0.87	0.95	0.98	0.95	0.96	0.93	0.93	0.95	0.95	0.96	0.96
(1, 0, 0, 0)	0.78	0.79	0.56	0.80	0.88	0.96	0.62	0.89	0.93	0.98	0.75	0.92
(0, 1, 1, 1)	0.88	0.73	0.53	0.42	0.87	0.84	0.03	0.03	0.71	0.70	0.96	0.96
total	0.88	0.81	0.766	0.781	0.934	0.941	0.827	0.857	0.939	0.943	0.942	0.961

Table 2: accuracy per state

6 Discussion

In this section we discuss the success of the method, its generalisability, and future research.

6.1 Evaluating BNs over all possible evidence states using simulations

With this method, we can show that we can create BNs that predict output probabilities that correspond to the limiting frequencies of events in the simulation. We see that both the preference ordering as well as the probabilities in the output nodes in Table 4 are near identical to those in Table 8. This means that the best BN reflects the frequencies in the simulation well, not just on one set of cumulative evidence, but on nearly all sets of cumulative evidence.

The only evidence set where the BN does not reflect the ground truth, is for the evidence set of (0, 1, 1, 0). This is a situation where the agent is seen stealing, but the object is not gone. In the test data of 10,000 entries, there is never an entry where both stealing and dropping are true. Hence, we find that the BN does not predict the output correctly in this case.

When less data is available to the K2 algorithm, such as the case where the simulation was run for 10 times (Table 3), we find that the accuracy of the network decreases. This means that the network is less able to predict the right outcome across all possible evidence states.

This method also allows us to see when the BN is assigning probability values to mutually exclusive nodes that violate probabilistic constraints. The joint probability over all three states (steal, drop, neither) can never be greater than 1, which means that none of the probability values of steal, drop, and neither, can be smaller than 0. However, in both Table 3, and Table 4, we find that P('neither') can sometimes be smaller than 0. This is also visible in the cumulative probability plots (see Figure **??** where for E_camera_seen_stealing_1_0, the joint probability of outcomes > 1). This shows that the mutually exclusive relation between stealing and dropping is not learned correctly in the network. This improves (but is not gone), when the K2 algorithm has access to more data.

This method of evaluating a BN over all states of cumulative evidence can distinguish between better and worse BNs, shows when a BN goes wrong and gives us insight in why that happens. Modellers of data-poor BNs are might think about creating an expected preference ordering on all possible evidence states, before starting to build and test the BN, in order to check whether their BN aligns with their common-sense predictions. If the BN is wrong, this is a sign that either the modeller's estimation is wrong, or that the probabilities as entered in the BN are not correctly representing your actual beliefs.

6.2 Can a modeller do this?

restructure and rewrite for language

We have shown that we can generate Bayesian Networks and evaluate them over all possible evidences states automatically, if we use the data as generated by the simulation. However, the real problem is, if a modeller could actually attempt to create a BN by hand without access to the underlying frequencies from the simulation. If we have a manually-crafted BN, but we also have the underlying true joint probability distribution over all variables in the simulation, we can test if this method of evaluation can show problems with a manually constructed Bayesian Network.

In order to do this, we created a network manually. Method for creating the BN

We considered the three different scenarios: 1) the object was stolen, 2) the object was dropped, 3) neither happened. Since we use the same nodes as in the simulation, we do not have a separate node for scenario 3, or a constraint node. Scenario 3 is represented by both other scenarios having low probabilities.

1. structure of the scenarios: The other scenarios are 2: [object dropped], and 1: [motive, sneak, stealing]. Within scenarios, we create temporal connections,

hence we only identify [motive → sneak → stealing] for now.

2.) structure between scenarios. Using the causality thing identified earlier, we place 'object dropped' scn2 before scn1. Resulting in the network [object dropped → motive → sneak → stealing]

3. Adding the evidence nodes. For each node, we consider causality. Psych report relates to motive only, object gone to both stealing and dropped, then camera sees stealing only directed to stealing node, with as parent 'thief seen on camera'. These reflect the causality in the situation.

4. For each node, create CPTs. To simulate the idea that we do not have exact probabilities (from the simulation), we place restrictions on the probabilities that we can add to the CPTs. We can use [1, 0, 0.99, 0.01, 0.25, 0.75] only in the CPT. These correspond to: Totally sure, direct causal relation (1, 0), pretty sure but there might be some reason why not (0.99, 0.01), and we are not sure but it's leaning in a certain direction (0.25, 0.75).

evidence	accuracy steal	accuracy dropped	
(0, 0, 0, 1)	0.46	0.46	0.33
(0, 0, 0, 0)	1.00	1.00	1
(0, 1, 0, 1)	0.99	0.99	1
(1, 0, 0, 1)	0.93	0.93	1
(0, 1, 0, 0)	1.00	1.00	1
(1, 1, 0, 1)	-0.93	-0.07	0
(1, 1, 1, 1)	0.95	0.95	1
(1, 0, 0, 0)	1.00	1.00	1
(0, 1, 1, 1)	1.00	0.99	1
total	0.711	0.806	0.815

Table 3: Accuracy in table

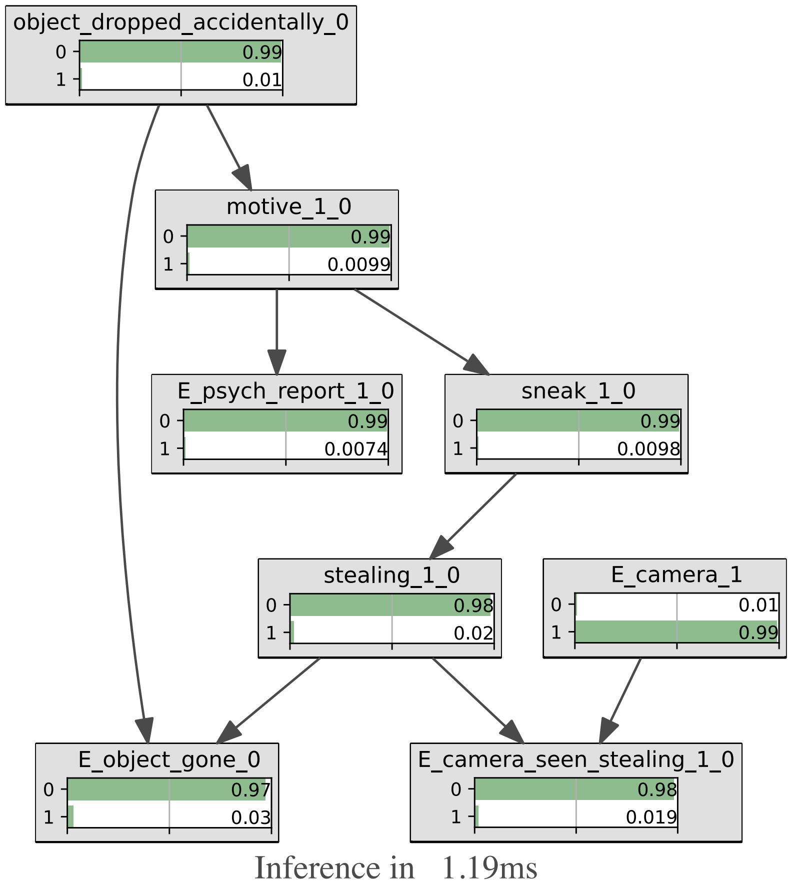


Figure 8: Manual network

evidence	H1	H2	H3
(0, 0, 0, 1)	P(steal) (0.55)	P(dropped) (0.45)	P(neither) (0.00)
(0, 0, 0, 0)	P(neither) (1.00)	P(steal) (0.00)	P(dropped) (0.00)
(0, 1, 0, 1)	P(dropped) (1.00)	P(steal) (0.00)	P(neither) (0.00)
(1, 0, 0, 1)	P(steal) (1.00)	P(dropped) (0.00)	P(neither) (0.00)
(0, 1, 0, 0)	P(neither) (1.00)	P(steal) (0.00)	P(dropped) (0.00)
(1, 1, 0, 1)	P(neither) (3.00)	P(steal) (-1.00)	P(dropped) (-1.00)
(1, 1, 1, 1)	P(steal) (1.00)	P(dropped) (0.00)	P(neither) (0.00)
(1, 0, 0, 0)	P(neither) (1.00)	P(steal) (0.00)	P(dropped) (0.00)
(0, 1, 1, 1)	P(steal) (1.00)	P(dropped) (0.01)	P(neither) (-0.01)

Table 4: Preference ordering with numbers for manual network

evidence	H1	H2	H3
(0, 0, 0, 1)	F(dropped) (0.99)	F(steal) (0.01)	F(neither) (0.00)
(0, 0, 0, 0)	F(neither) (1.00)	F(steal) (0.00)	F(dropped) (0.00)
(0, 1, 0, 1)	F(dropped) (0.99)	F(steal) (0.01)	F(neither) (0.00)
(1, 0, 0, 1)	F(steal) (0.93)	F(dropped) (0.07)	F(neither) (0.00)
(0, 1, 0, 0)	F(neither) (1.00)	F(steal) (0.00)	F(dropped) (0.00)
(1, 1, 0, 1)	F(steal) (0.93)	F(dropped) (0.07)	F(neither) (0.00)
(1, 1, 1, 1)	F(steal) (0.95)	F(dropped) (0.05)	F(neither) (0.00)
(1, 0, 0, 0)	F(neither) (1.00)	F(steal) (0.00)	F(dropped) (0.00)
(0, 1, 1, 1)	F(steal) (1.00)	F(dropped) (0.00)	F(neither) (0.00)

Table 5: Preference ordering with numbers for all evidence states on 'ground truth' for runs=10,000

Diagnosing problems: - overall accuracy: This network is about as accurate as the K2 network based on 100 runs, which is not the best but also not bad. Pretty good accuracy over all states for a network that was subjectively estimated and connected. Overall accuracy of network is

which is worse overall than the worst automatic network, however, it is about similar in performance? Seems reasonable.

- When we compare the performance of this network to the actual frequencies from the simulation, we see that this handcrafted network performs well. In all cases except 1 (which we'll get to later), the manual network predicts the same preference ordering over the outcomes as they happen in the grounded evidence states. - The manual network does not predict the exact probabilities from the grounded frequencies. Sometimes it is overconfident (such as in (0101) (1001)), other times it is underconfident (0001)). Analysis why here. (0, 0, 0, 1) - One big problem: 1101 state: modelled in network as if impossible (hence, -1 -1 probabilities). The simulation itself says that it is always stealing. weird. It means that the agent steals but is not shown on camera.

Conclusion: The manual network is quite good. We can identify problems with the network. Method works!

Implications, plausibility: - Even with a low degree of precision, if we have a conditional structure and a clear idea of what happens, the network is ok. However, in real life, this is not as obvious. The conditional structure that we can find here in the network makes it easy to elicit and use probabilities, but if we would try to apply this to a more complicated situation with more unknowns, we might not be able to find the probabilities. The events in the simulation are observable and clearly defined, in real life, we do not have the clear definition: we might not know what causes something else.

Problems:

- validity of the manual network: these probabilities in the network were subjectively estimator by the person who created the simulation. The probabilities were 'subjectively' estimated, which means that I did not look at, or calculated, probabilities to put in the network. However, I've been working on this project for 3/4 year so it is likely that I have a pretty good feel for what the probabilities in the network should approximate, and I know the causal structure of the simulation. Therefore, it is likely that people who just have access to the simulation output, will create 'worse' networks. However, this can be detected by the evaluation method, hence, the evaluation network works. Perhaps if the user had the chance to interact with the simulation, they might be able to estimate some frequencies at which events happen.

- if the simulation was more difficult, the automated versions would continue to do well, but the manual approach would flouder. Size complexity

Hence modellers could estimate probabilites and create nice networks but probably not in real complex life.

6.3 Evaluating BNs over all possible evidence states in real life

In this section we evaluate two Bayesian Networks with the method outlined before. rewrite for language

6.3.1 de Zoete so-called

The first is a network by de Zoete et all, about child abuse. This is from a paper about probabilistic paradoxes in legal reasoning. IT considers a case where we want to know if a child is abused. We have a certain behaviour that is associated with abuse, but some study shows that this behaviour occurs about as frequently in children who are not abused. Only when the behaviour occurs together with an additional, different behaviour, there's a strong probability of abuse. It has been proposed that the combination of two evidence pieces is difficult to model - however, de Zoete at all find a way to model it, using the Bayesian Network presented in Figure **??**. In this network, they divide the potential children into three groups: 1) a child who is abused, who must have both the behaviour and the additional characteristic, 2) a child who is not abused, who has the behaviour but can never have the additional characteristic, and 3) a child who is not abused, who does not have the behaviour, and also does not have the additional characteristic.

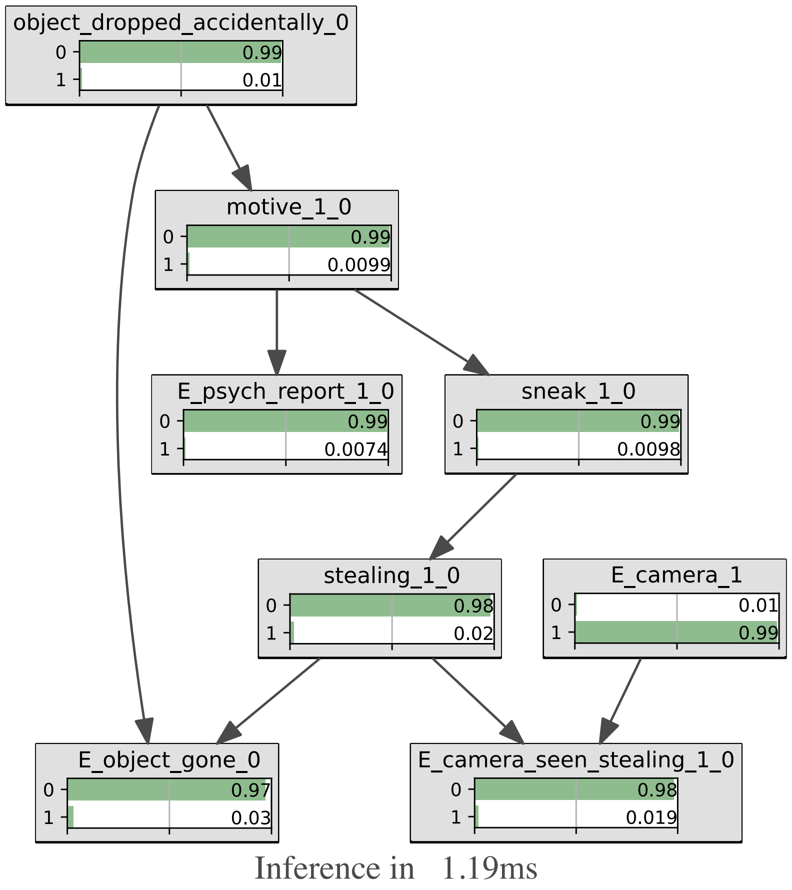


Figure 9: Manual network

In the paper, they present a table with some possible outcomes, to show that this ‘both things must be true’ thing is correct. However, using the method in this paper, we can evaluate over all evidence states. Here, we have two pieces of evidence, exhibiting behaviour (B) and additional characteristics (C). We consider all combinations of evidence in Table 9. The network performs well in 3 cases (4 if you count the ‘no evidence’). In case of both pieces of evidence are true, the posterior of abuse

Evidence	P(abused)	p(nonAbusedBehaviour)	p(nonAbusednotBehaviour)
no evidence set	0.40	0.50	0.10
B and C	1.00	0	0
B and ¬C	0.375	0.625	0
¬B and *C*	0.25	0	0.75
¬B and ¬C	0.4286	0.4762	0.0952

Table 6: Preference ordering with numbers for all evidence states on 'ground truth' for runs=10,000

is 1. If one of the pieces of evidence is false, the posterior for abuse drops compared to its prior.

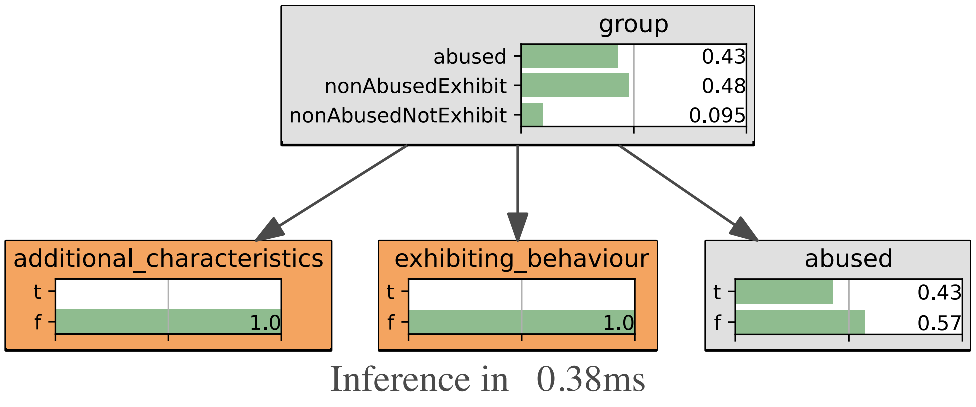


Figure 10: Manual network

6.3.2 van Leeuwen 2019

We applied this method to the network presented in van Leeuwen (2019), shown in Figure 11. This network models a crime case where a suspect P murdered a tenant, then fled by car, and later called his parents to confess the crime, which is modelled as scenario 1 (scn1). In his own testimony, P states that he has amnesia and does not remember the day of the murder, and also that he was kidnapped, which is modelled as scenario 2 (scn2). The two scenarios are mutually exclusive and exhaustive. Evidence node names are shortened in table for convenience: 'testimony_amnesia' (TA), 'Medical_investigation_found_no_amnesia' (MA), 'testimony_kidnapping' (TK), 'no_concrete_evidence_for_kidnapping' (NK), 'phonecall_parents'

(PP), ‘car_with_bloodstains_found’ (CB), ‘body_found’ (BF), ‘signs_of_violence’ (SV), ‘phonecall_with_friend’(PF), ‘testimony_conflict’ (TC), ‘weapon_found’ (WF).

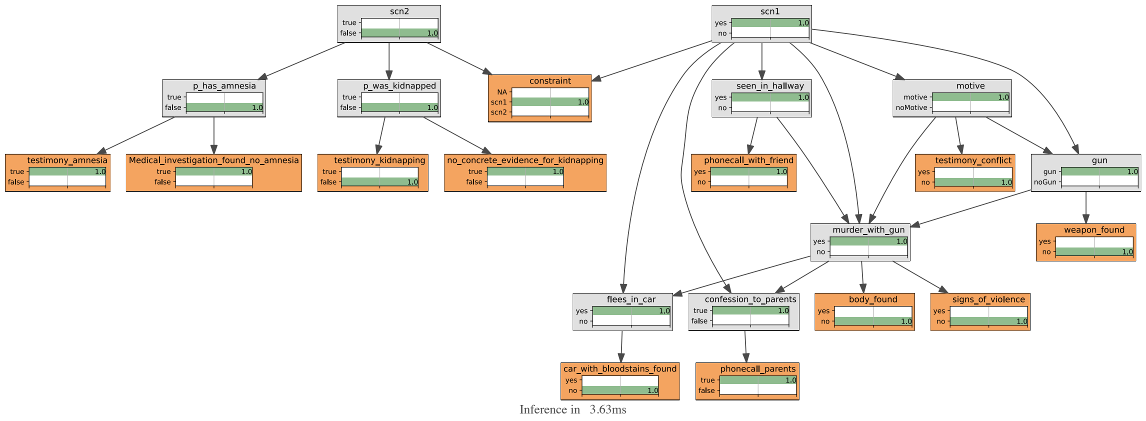


Figure 11: Example of subjective probability estimation resulting in a guilty verdict for insufficient evidence

This network has 11 evidence nodes, which means a potential of $2^{1}1 = 2048$ total combinations of evidence. Time-technically, this took 7.66 seconds. However, the network constrained some combinations of evidence states, in 56.25% of evidence states, the combination of evidence was not allowed by virtue of the BN. In the remaining 896 evidence states (see Appendix), the evidence state was applied to the network and the posterior probability for scenario 1 and scenario 2 were calculated.

In general, we find that scenario 1 and scenario 2 are about equally preferred under the states of evidence, in 428 states, scenario 1 has a higher posterior probability, and in 468 states, scenario 2 has a higher posterior. In 768 states of evidence, it is the case that either scenario 1 or scenario 2 is equal to 1, which means that we are fully convinced that that scenario is correct: in half the cases, we fully believe scenario 1, and in the other half, we fully believe scenario 2.

We do not have an automated baseline for this network so that means that if we wanted to assess the validity of the BN, we would have to create a baseline by hand. This would mean that we have to write down our predictions for posteriors or preference ordering for every combination of evidence by hand, and check them against the outcome of the BN. This is very demanding, confusing, and boring. Hence, instead of fully performing the method outlined in this paper, we only show some interesting highlights of where we can see that the network goes wrong, or does something right. These highlights are extracted from the table in the Appendix and placed in Table 10 (number corresponds to number in Table 10).

	evidence	H1	H2
1	TA, MA, TK, NK, CB, PP, BF, SV, PF, TC, WF	scn1 (1.00)	scn2 (0.00)
2	TA, ¬MA, ¬TK, ¬NK, ¬CB, PP, ¬BF, ¬SV, PF, ¬TC, ¬WF	scn1 (1.00)	scn2 (0.00)
3	TA, MA, TK, NK, CB, PP, ¬BF, ¬SV, PF, ¬TC, ¬WF	scn1 (0.00)	scn2 (1.00)
4	TA, MA, TK, NK, CB, PP, BF, ¬SV, PF, ¬TC, ¬WF	scn1 (0.40)	scn2 (0.60)
5	TA, ¬MA, TK, ¬NK, ¬CB, PP, BF, ¬SV, PF, TC, WF	scn1 (0.07)	scn2 (0.93)
6	¬TA, ¬MA, ¬TK, ¬NK, ¬CB, ¬PP, ¬BF, ¬SV, ¬PF, ¬TC, ¬WF	None	None

Table 7: Interesting rows extracted from complete table

1. We find that, if all evidence nodes are true, which is the situation as found in the court case, the network shows that scenario 1 is the case, and that the suspect is guilty, with a probability of 1. This is the expected output, and corresponds to the outcome of van Leeuwen (2019).

2. There is no body, no signs of violence, no car with bloodstains, no evidence of existing conflict. The only evidence that the prosecution has, are two phone calls. The outcome of the network is that the suspect is guilty (P(scn1 = 'yes') = 100%). But the evidence that is entered into this network, would not reasonably support this conclusion. No reasonable arguer would argue that this state of evidence is sufficient for a guilty verdict.

3. Here we fully believe scenario 2, so that the suspect is not guilty. However, all evidence that we have learned about scenario 2 shows that scenario 2 is not true: we have testimony for amnesia, but no medical record of it. We have testimony for kidnapping, but no concrete evidence for kidnapping. This should mean that our posterior belief in scenario 2 should decrease. We also have a little evidence on the side of scenario 1: two phone calls, and a car with bloodstains. This would not be enough for a conviction, but it should be enough that our belief in scenario 1 is not 0.

4. The same case as evidence state in case 3, but now we change our evidence. From not having a body found, we add that we find the body, and we see that our belief in scenario 1 increases, from 0 to 0.40. This is a change of our belief in a reasonable direction given new evidence. The size of the belief-change, from 0 to 0.40, is difficult to assess without a realistic baseline.

5. Here we find that the suspect says that he has amnesia and also that he was

kidnapped, and this is supported by the other evidence. This adds support for scenario 2. However, we also find a lot of support for scenario 1: a body, a testimony of conflict, the phone calls, weapon found. Just not the car, and no signs of violence on the scene. Here, my subjective probability estimation machine thinks that, even though scenario 2 has some good evidence going for it, I believe that scenario 1 is more likely than 0.07. Hence, the output does not line up with my expectation.

6. This is a special case: if all evidence nodes are false, the inference engine shows the result that the evidence is altogether incompatible. Since both scenarios are mutually exclusive and exhaustive, this result is sensible and shows that the BN can indeed structure our reasoning.

Even without using a simulation, this method can be used to gain insight in whether the BN conforms to our expectations. We can identify cases where it seems to work correctly, such as case 1) and case 3). In case 1), the output conforms to the one datapoint that we have (given all evidence, the suspect is guilty), the BN makes the same decision as the court. In case 4), we find that the network, in this circumstance, responds sensibly to a change in evidence. However, in case 2), 3) and 5), we find the performance of the network not convincing. In case 2), there is insufficient evidence for scenario 1, yet the network chooses scenario 1. In case 3), there is insufficient evidence for scenario 2, yet the network chooses scenario 2. In case 5), there is support for both scenarios, but the posterior probability for scenario 1 seems (subjectively) too low. Furthermore, we now also know which states are deemed impossible (example: case 6). However, since we do not have a simulation or real-life test-cases, we do not have a way of 'objectively' checking whether the network performs correctly or not, and to objectively identify combinations of evidence for which the network is performing badly. We can only do spot-checks and manually search through the output to see if we found anything interesting. Also, if we found a case for which the network performs incorrectly, it is also not clear what we should do with that: how should we change the probabilities in the CPT in order to increase 'accuracy'?

It is possible to apply this method of evaluation to other networks (given that they are produced in an open-access format), to non-systematically check the performance of these networks over all possible evidence valuations. In the networks as considered in the Anjum case study in Vlek (2016), or the Simonshaven case in Fenton et al. (2019), the evidence nodes are distinguished from the hypothesis nodes. Even without considering the temporality of evidence, we can set the cumulative evidence all at once. However, since we do not have simulations for grounding, or data for grounding, we cannot actually evaluate the networks based on 'objective' measures. Manually considering all evidence sets will be even more problematic than before. Instead of 11 nodes, the Simonshaven network has 13 evidence nodes, and the Anjum network has 20 evidence nodes. If we assume that calculating 1 case will take around 0.0037 seconds (as calculated from 7.66 seconds for 2048 entries, this does not hold entirely because it also depends on the number of hypothesis nodes), producing all possible Simonshaven states and posteriors should take $2^{13} = 30$ seconds, which is still reasonable, but the Anjum network would take 64 minutes. That is only considering the automatic calculation of the posteriors, and does not consider the time-complexity of creating a manual or automatic baseline. Additionally, some of the nodes in the Simonshaven are not binary nodes, resulting in a higher complexity as well.

6.4 Problems

There are three main problems with this approach that limit its generalisability. These problems are the combinatorial explosion of the evidence states, the implausibilities of finding proper grounding, even in simulation, specifying more than we actually need, and the imprecision of the BN definitions of the random variables in real life.

1. For this approach, we need to consider all possible evidence states. In the worst case, if we have e evidence states, this means that we have e^2 possible evidence states. We would have to define a preference ordering or probability distribution on the output nodes in 2^e cases. This is plausible in a network with 4 evidence nodes. However, as we saw in the discussion of the literature, this becomes implausible when there are many evidence nodes. This necessarily puts an upper bound on the level of granularity one can model in their network. If our network becomes more detailed in how evidence is represented, the higher the probability that considering all possible evidence states becomes improbable. A rough estimation shows that a 20-evidence-node network will take about an hour, but 20 pieces of evidence are not much in a complex case. Due to exponentiality, a network with 25 evidence nodes would result in 2.3 months of calculations. There's also the combinatorial explosion, where you would need to define many more cases of evidence than you are actually reasoning with. However, we are already doing this implicitly by using a Bayesian Network for reasoning in the first place.
2. Due to the simulation approach in this paper, we can establish a statistical

ground truth in step 4) of our method. We can create both a joint probability distribution and a preference ordering automatically from the simulation. In real life, this would have to be elicited in some way. Even if we assume that we are not in a combinatorial explosion, we would still need to define either a subjective probability distribution over all possible outcomes, or a preference ordering, and use this as a baseline to test the predictions of our BN against. This involves subjectivity twice: first in defining the preference ordering or probability distribution over outcomes, and second the probabilities that are inputted in the CPTs. Future research might establish how successful people are in estimating preference orderings or probability distributions over output nodes, so that we can use these subjective orderings as a baseline, instead of an ordering generated by frequency data from the simulation.

As we saw when we applied the method to the network without a simulation, it is not plausible to create a manual baseline, as this would be mind-numbingly boring and consume too much time. Creating a simulation would mean that the simulation should go beyond one room, and instead model an entire city, all places that the suspect can be all the time. This requires a lot more data and attention than is possible now, and it is also unclear how we would test the validity of the simulation, which would result in all our problems just being shifted up a step (to the simulation level, instead of in the BN). This means that simulations might be a good tool to investigate the methods of BNs, but are not suitable for actually investigating real-life crimes (although they might be usable as a tool for predicting situations - stuff where we have frequency data about.

Yes, simulations are a nicer modelling primitive than straight BNs. Could use a true simulation to generate data and cases and train a BN on that, and use decision making based on BN data. A simulation like programmed now is nice as a data-generator, but not so much as a reasoning thing. However, it is unclear unknown difficult to know in what detail one should model for a real case, instead of just a test-case.

Simulations are nice because you do not have to deal with calculating the probabilities yourself. For instance, calculating the probability of motive, means that we need to calculate vulnerable and valuable. This can be done, given that the guy has a target: valuable = $(2/5)*(1/2)*(1/2) = 0.1$, while vulnerable = $(1/5)+(3/5)(1/2) = 0.5$. Hence, motive = 0.05, or 5%, just by bare probability calculations, which matches the probability as found in the network and the training dataset. However, this was relatively easy because these were straightforward uniform distributions. This calculation is going to be more complex if we're dealing with either more complex distributions (normals, etc.), or if we do not have a good idea about the underlying distributions. However, doing this using simulations makes it a lot easier, the numbers 'just' emerge without any calculation. We even encounter this in the simulation right now: the probability that the camera sees the stealing is very difficult to calculate analytically. Simulations are a more natural way to think about this.The simulation can also easily become more relevant (more like a level 1 or 2 simulation in the taxonomy), by creating more realistic agent motivation models and behaviour. On the other hand, debugging simulation results is really annoying.

3. Too much data: in the court case, we are only considering the evidence that we believe is true. We do not have to consider these counterfactual states of evidence, all 895 of them. In a BN, these other 895 cases are 1) specified and 2) must be correct. Isn't this asking too much? Other approaches to reasoning, scenario-based, argumentative or hybrid do not need to fill out the CPTs, there they only need to consider the evidence by itself. explain

4. More generally, a problem with BNs is that a random variable, which is a function that maps a set of events to a 'measurable space'. Informally, in this case, the RV maps an event to either T or F. There are two problems: how we define the set of events that we are mapping (domain), and the validity of the 'mapping' function.

   (a) **Defining the domain, or the reference class problem:** Defining the set of events is inherently problematic, and is called 'the problem of the reference class' (Colyvan et al., 2001). The problem of the reference class is in picking the set of events that belong in the domain. This problem is relevant in both the simulation and the real world. However it is not as relevant in the simulation because often there is only one interesting/relevant way to interpret the events (eg there are not two equally "good" 'reference classes' for a node like 'sneak_1_0'). There is only one relevant class (which is defined with respect to the agent behaviour), and that is the relevant reference class. However, in real life, you cannot know which reference class is the most 'relevant' or fair to choose. It is unclear how to get out of this bind, but we know that we do reason pragmatically and informally with the reference class.

   (b) **Defining the mapping, and its validity:** This is no problem in the simulation, we know what the methods for determining the probabilities in the nodes are valid (which means that they always correctly reflect the underlying progress that is generating them), because it's literally programmed in this way - directly reflecting the states, we literally just count the frequencies. However, in real life, we might not even know the validity of different possible mappings for the same node. In short, how do we determine in real life that 'sneak_1_0'? How can we determine if anyone is actually 'sneaking', and how will we know if this method of determination is valid? This is also why BNs for forensic-statistical BNs are more legitimate: in these forensic-statistic fields, we have defined clear methods and we know the method through which we are mapping: these variables are operationalised. This is not the case for more 'vague' nodes.

7 Conclusion

7.0.1 Summary

We have shown that simulations can be used to ground Bayesian Networks and we created an evaluation method for them. Bayesian Networks can be created from the simulation automatically, and these Bayesian Networks can be evaluated for goodness systematically. We know that these networks are accurate because 1) the predicted prior probabilities of events in the network correspond exactly to the frequency of events in the simulation and 2) adding all possible states of evidence to the Bayesian Network results in the same output prediction as given by the simulation. Accuracy increases the more data is available to automatically construct the network, hence we find that some networks are better than others, and we can make a real evaluation.

However, this method of evaluation is not suitable for 1) real-life criminal cases and 2) networks with > 25 evidence nodes. In real-life criminal cases, we do not have the grounding necessary to check whether the network is accurate automatically. Creating this checking-dataset by hand means that we have to specify for all possible evidence states what our prediction of the output nodes is, which takes too long and we do not have a good grounding to prefer one set of evidence states over the other. Additionally, in real life we do not have to reason over all sets of possible evidence, we just need to reason with the set of evidence that we found. There need to be fewer than 25 evidence nodes in the network (given personal-computer constraints), as runtime increases exponentially with the number of nodes.

7.0.2 Future Research

The simulation in this paper is not complex, and does not model reality in any realistic way. The purpose of this simulation was just to generate data, and did not need to reflect any specific facts about the world. However, agent-based modelling can be useful for investigating different aspects of Bayesian Networks, or reasoning with evidence more generally. Increasing the level of the simulation from level 0 to something that reflects reality is useful to investigate problems such as the Opportunity Prior as presented in Fenton et al. (2017).

Letting people estimate probabilities on simulations to investigate elicitation accuracy.

Bibliography

Bench-Capon, T. (2019). Before and after dung: Argumentation in ai and law. *Argument and Computation*, 11:1–18.

Chen, X.-W., Anantha, G., and Lin, X. (2008). Improving bayesian network structure learning with mutual information-based node ordering in the k2 algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):628–640.

Colyvan, M., Regan, H. M., and Ferson, S. (2001). Is it a crime to belong to a reference class. *Journal of Political Philosophy*, 9(2):168–181.

Cooper, G. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

Dahlman, C. (2020). De-biasing legal fact-finders with bayesian thinking. *Topics in Cognitive Science*, 12(4):1115–1131.

Dawid, A. P. (2008). Beware of the dag! In *JMLR: Workshop and Conference Proceedings 6: 59–86*.

de Koeijer, J. A., Sjerps, M. J., Vergeer, P., and Berger, C. E. (2020). Combining evidence in complex cases-a practical approach to interdisciplinary casework. *Science & Justice*, 60(1):20–29.

Di Bello, M. and Verheij, B. (2018). Evidential reasoning. In Bongiovanni, G., Postema, G., Rotolo, A., Sartor, G., Valentini, C., and Walton, D. N., editors, *Handbook of Legal Reasoning and Argumentation*, pages 447–493. Springer, Dordrecht.

Ducamp, G., Gonzales, C., and Wuillemin, P.-H. (2020). aGrUM/pyAgrum : a Toolbox to Build Models and Algorithms for Probabilistic Graphical Models in Python. In *10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 609–612, Skørping, Denmark.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357.

Fenton, N., Lagnado, D., Dahlman, C., and Neil, M. (2017). The opportunity prior: A simple and practical solution to the prior probability problem for legal cases. In *ICAIL '17*. ICAIL.

Fenton, N., Neil, M., and Lagnado, D. A. (2012). A general structure for legal arguments about evidence using bayesian networks. *Cognitive Science*, 37(1):61–102.

Fenton, N., Neil, M., Yet, B., and Lagnado, D. (2019). Analyzing the simonshaven case using bayesian networks. *Topics in Cognitive Science*, 12(4):1092–1114.

Gerritsen, C. (2015). Agent-based modelling as a research tool for criminological research. *Crime Science*, 4(1).

Gilbert, N. and Terna, P. (2000). How to build and use agent-based models in social science. *Mind & Society*, 1(1):57–72.

Gilbert, N. and Troitzsch, K. (2005). *Simulation for the social scientist*. McGraw-Hill Education (UK).

Hamilton, C. S. and Pollino, C. (2012). Good practice in bayesian network modelling. *Environmental Modelling and Software*, 37:134–145.

Kadane, J. B. and Schum, D. A. (1996). *A Probabilistic Analysis of the Sacco and Vanzetti Evidence*. Wiley, Chichester.

Kazil, J., Masad, D., and Crooks, A. (2020). Utilizing python for agent-based modeling: The mesa framework. In Thomson, R., Bisgin, H., Dancy, C., Hyder, A., and Hussain, M., editors, *Social, Cultural, and Behavioral Modeling*, pages 308–317, Cham. Springer International Publishing.

Meester, R. and Slooten, K. (2021). *Probability and Forensic Evidence*. Cambridge University Press.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Pennington, N. and Hastie, R. (1993). Reasoning in explanation-based decision making. *Cognition*, 49(1–2):123–163.

Prakken, H., Wyner, A., Bench-Capon, T. J. M., and Atkinson, K. (2013). A formalization of argumentation schemes for legal case-based reasoning in ASPIC+. *Journal of Logic and Computation*, 25(5):1141–1166.

Schum, D. A. and Martin, A. W. (1982). Formal and empirical research on cascaded inference in jurisprudence. *Law & Soc'y Rev.*, 17:105.

Timmer, S. (2016). *Designing and Understanding Forensic Bayesian Networks using Argumentation.* PhD thesis, University of Utrecht.

Van Der Gaag, L. C., Renooij, S., and Coupé, V. M. (2007). Sensitivity analysis of probabilistic networks. In *Advances in probabilistic graphical models*, pages 103–124. Springer.

van Leeuwen, L. (2019). A comparison of two hybrid methods for analysing evidential reasoning. In *JURIX*.

Verheij, B., Bex, F., Timmer, S. T., Vlek, C. S., Meyer, J.-J. C., Renooij, S., and Prakken, H. (2015). Arguments, scenarios and probabilities: connections between three normative frameworks for evidential reasoning. *Law, Probability and Risk*, 15(1):35–70.

Vlek, C. (2016). *When stories and numbers meet in court.* PhD thesis, University of Groningen.

Wagenaar, W. A., Van Koppen, P. J., and Crombag, H. F. (1993). *Anchored narratives: The psychology of criminal evidence.* St Martin's Press.

Wieten, R., Bex, F., Renooij, S., and Prakken, H. (2019). Constructing bayesian network graphs from labeled arguments.

Wigmore, J. H. (1931). *The Principles of Judicial Proof or the Process of Proof as Given by Logic, Psychology, and General Experience, and Illustrated in Judicial Trials, 2nd ed.* Little, Brown and Company, Boston (Massachusetts).

Appendix

This appendix contains:

The probability distribution and preference ordering over outcomes for all possible evidence for the automated networks on the simulation.

The probability distribution and preference ordering over outcomes for all possible evidence for the Jurix 2019 network.

7.1 Tables accuracy automated network

7.2 Jurix

7.2.1 Network

```
%%%%% SCENARIO 2 %%%%%

potential (object_dropped_accidentally_0)

0          1
( 0.99 0.01 )


%%%%% SCENARIO 1 %%%%%

potential (motive_1_0 | object_dropped_accidentally_0)

0          1
( 0.99 0.01 ) %  object_dropped_accidentally_0=0
( 1           0 ) %  object_dropped_accidentally_0=1


potential (sneak_1_0 | motive_1_0)

0          1
( 1         0 ) %  motive_1_0=0
( 0.01 0.99 ) %  motive_1_0=1


potential (stealing_1_0 | sneak_1_0)

0          1
( 0.99 0.01 ) %  sneak_1_0=0
(0.01 0.99 ) %  sneak_1_0=1



%%%%% EVIDENCE NODES %%%%%%
```

```
potential (E_camera_1)
0         1
( 0.01 0.99 )



potential (E_psych_report_1_0 | motive_1_0)
0         1
( 1 0 ) %  motive_1_0=0
( 0.25  0.75 ) %  motive_1_0=1



potential (E_camera_seen_stealing_1_0 | E_camera_1 stealing_1_0)
0     1
( 1 0 ) %  E_camera_1=0  stealing_1_0=0
( 1 0 ) %  E_camera_1=0  stealing_1_0=1
( 1 0 ) %  E_camera_1=1  stealing_1_0=0
( 0 1 ) %  E_camera_1=1  stealing_1_0=1




potential (E_object_gone_0 | object_dropped_accidentally_0 stealing_1_0)
0     1
( 1 0 ) %  object_dropped_accidentally_0=0  stealing_1_0=0
( 0 1 ) %  object_dropped_accidentally_0=0  stealing_1_0=1
( 0 1 ) %  object_dropped_accidentally_0=1  stealing_1_0=0
( 0 1 ) %  object_dropped_accidentally_0=1  stealing_1_0=1
```

7.2.2 Table