Evaluating Bayesian Networks from Multi-Agent Simulations.

Ludi van Leeuwen

July 15, 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?

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 (Pearl, 1988). 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 (Dawid, 2008).

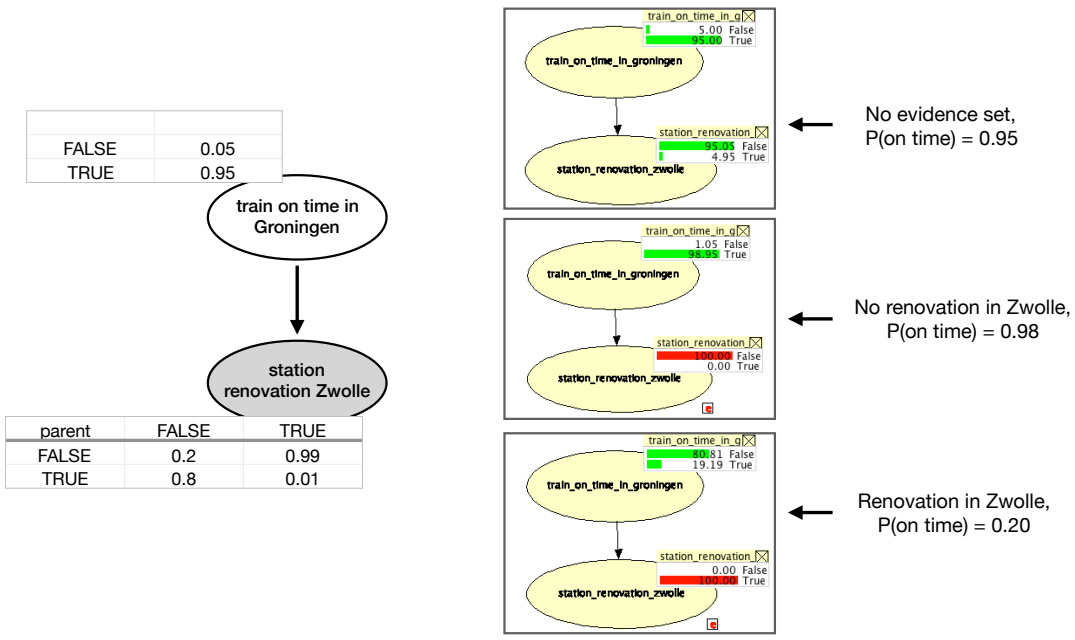


Figure 1.1: An example of a Bayesian Network, with and without evidence set.

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.

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!

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.

Our 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?**

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 (Vlek et al., 2015), 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 (Verheij et al., 2015). 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 (Pennington and Hastie, 1993), (Wagenaar et al., 1993), 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: (Bex, 2010) and (Timmer, 2016). This project does not consider the argumentative approach, and is mainly based on the work of Vlek (Vlek et al., 2015), (Vlek, 2016) and Fenton (Fenton et al., 2012), (Fenton et al., 2019). 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 (Dahlman, 2020).

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 (J Kadane, 1996), (Fenton et al., 2019), cases from fiction (Fenton et al., 2012) or fictionalized crime cases (van Leeuwen, 2019). In situations where the network represents aspects, the nets model DNA or blood-spatter evidence, for methods see (Ronald Meester, 2021).

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 (Ducamp et al., 2020), 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 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'.

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 (Allen and Pardo, 2007), (Colyvan et al., 2001)). Probabilities can be pure frequencies, or can be subjectively elicited from experts (Renooij, 2001), (Druzdzel and van der Gaag, 2000).

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 (Kazil et al., 2020).

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 (Bosse and Gerritsen, 2008), theories of behaviour (Gerritsen, 2015), and police strategies in urban crime (Haojie Zhu, 2021). 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 (Haojie Zhu, 2021).

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 (Colyvan, 2013), 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

Creating Simulations to Evaluate Bayesian Networks.

3.1 Introduction

This chapter lays out the general method for creating simulations with automatic Bayesian Networks, as well as how to evaluate these Bayesian Networks. The general process of creating simulations to evaluate Bayesian Networks is illustrated in Figure 3.1. We start by defining a simulation with agents, and then have reporters report on that simulation in the 'Experiment' stage. Relevant events in simulations are collected by the reporters in each run. The collection of runs is used by an automatic Bayesian Network constructor algorithm to construct a Bayesian Network automatically in the 'Building BN' stage. Finally, the constructed Bayesian Network is evaluated with respect to the criteria in the 'Evaluate BN' stage. These three stages are explained in this section. Specific instances of simulations and networks created with this process are the subject of the next chapters.

3.2 Setting-up an Experiment

An experiment consists of a simulation and reporters. Reporters are defined separately from the simulation because they are not inherent to it - they are defined with respect to what we want to know about the simulation.

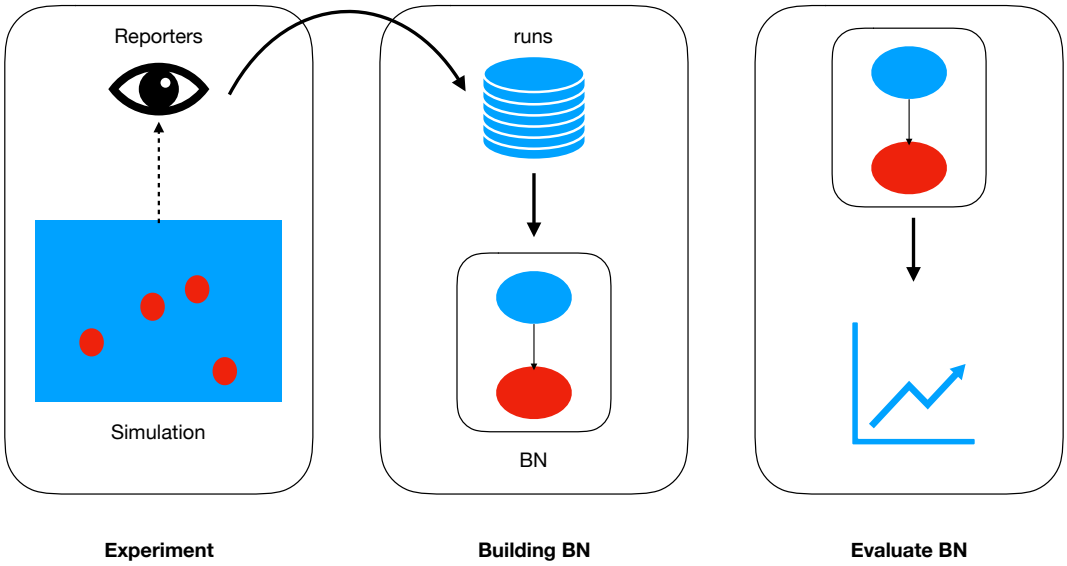


Figure 3.1: Method for evaluating automatically generated Bayesian Networks from simulations.

3.2.1 Simulation

Structure of a simulation

In the simulation, we are simulating some sort of criminal scenario - a theft, usually. The simulation can be as precise as necessary, but there are certain things that need to be present: we need to have states that can occur and there needs to be evidence for those states. The granularity of the simulation and its complexity depend on the modeller and her requirements.

Spatial and Non-spatial simulations.

This thesis uses two types of simulations: spatial simulations and non-spatial simulations.

The non-spatial simulation is not really a multi-agent system. It is only a forward-chaining algorithm that represents world-states. In non-spatial simulations, all agent states are essentially brought about by a combination of randomly generated numbers, and reasoning rules. For example, if an agent has a tendency to lie (randomly generated), and it has the opportunity for lying (brought about due to the current non-spatial simulation), then it

will lie. Hence a combination of behavioural rules and randomly generated numbers results in a state of 'agent lies' of 1.

In a spatial simulation, an agent's behaviour is mediated with respect to their environment - agents cannot pass through buildings, or cannot see agents that are far away, or can only steal from another agent when that agent is nearby. Interactions and behavioural rules and randomly generated numbers are all brought together: if an agent is near another agent (chance interaction), and it has a tendency to steal (randomly generated), then it will attempt (but might not succeed) in stealing. Here the behavioural rule might lead to a more interesting/complex/complicated outcome than in the non-spatial simulation.

For this project, the spatial simulations are more complex and interesting than non-spatial simulations, as there are more possibilities for variety.

3.2.2 Reporters (as Random Variables)

In the simulation, certain states can be brought about. For example, an agent can succeed in stealing an object, or in lying, or in having a motive (or in not having those things). We need a way to observe these states: this is where reporters come in. A reporter reports the outcome of a relevant event or state 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:

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

Not all the states in the simulation have a reporter associated with it - otherwise there would be infinitely many reporters. There could have reporters for for $agent\_Q\_at\_x\_1\_y\_200$. Hence, there is subjectivity involved in creating and selecting the relevant reporters.

The global state of one run of the simulation, is the combination of all reporters.

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

or,

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

for n reporters.

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:

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

3.3 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 (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. In other domains, it can be difficult to find one of the correct orderings. However, in the network as created from the simulation, we can automatically add the appropriate ordering by collecting the temporal ordering of hypothesis (non-evidence!) events in the simulation. The evidence nodes are added at the end, to ensure that evidence nodes are never parents to hypothesis nodes.

For example, in a hypothetical simulation with 3 hypothesis nodes {'going_cycling', 'cycling_over_icy_patch', 'slipped'} and 2 evidence nodes {'street_looks_shiny', 'bruised_leg'}, even though the total temporal ordering of events in the simulation would be: {'going_cycling', 'street_looks_shiny', 'cycling_over_icy_patch', 'slipped', 'bruised_leg'}, the ordering as used

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

in our experiments would be: {'going_cycling', 'cycling_over_icy_patch', 'slipped', 'street_looks_shiny', 'bruised_leg'}, so that we can keep to the evidence-idiom construction as outlined in Fenton et al. (2012).

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.

After the algorithm generates the network, there is an adaptation that still must be made. The algorithm does not know how to handle impossible states, which are cases where combinations of valuations for parent nodes are impossible. For example, a thief having a motive to steal an object when the thief does not know that the object exists - this can never happen in the simulation. 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.

3.4 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: structural criteria, performance criteria and human criteria.

3.4.1 Structural Criteria

1. Hypotheses are ordered temporally.

2. Evidence connects to hypotheses.

3. Relevance: All relevant events are in the BN, all irrelevant events are outside of the BN.

4. Independent events are not connected to each other.

3.4.2 Performance Criteria

1. Accuracy.

   We generate a training set of m entries, by running the simulation respectively m. At every run, we collect the state of the simulation from the reporters. This results in m rows for the training data. The K2 algorithm is trained on the full m entries of the training set, which results in a Bayesian network.

   We calculate the accuracy over the set of all possible global states. There are some global states that are not allowed by the simulation. This would be equivalent to events that can never happen in the world (like a light being both off and on at the same time [2]). For all possible global states, we use them as input for our Bayesian Network. For each of the possible states, we set every evidence node to the value corresponding to its state in the Bayesian Network, then look at the posterior probability of the outcome node. Then the posterior probability is rounded to either 0 (if posterior ≤ 0.5) or to 1 (if posterior > 0.5), to represent making a decision and compared to the state of the outcome in the world state.

   If the outcome global state and the rounded posterior probability of the global state match, the network has correctly predicted the state of the outcome reporter in the simulation. If they do not match, the network has made an incorrect prediction. If we cannot add some evidence because that would cause the joint probability distribution to become incoherent, we count that as a failed prediction. The accuracy is calculated as

   $$\frac{correct}{total}$$

2. Root Mean Squared Error.

   Similar to accuracy, except that we do not round the posterior probability. We also calculate the root mean square error as a way to see how close the posterior of the network is to the state of the world. For each row compute

   $$\sqrt{\frac{\sum_i^N (state_i - posterior_i)^2}{N}}$$

[2] yes, you can probably think of some defeasible reason why it would be possible to have a light be off and on at the same time, but you'd need to model this at the simulation-level

3. Correspondence.

   To what extend do the probabilities found in the Bayesian Network correspond to the frequencies encountered in the simulation?

4. Sensitivity Values of Output Node.

   Performing meaningul sensitivity analysis on the network is difficult. In essence, a sensitivity analysis's goal is to see how small changes in a conditional probability table (cpt) of a node can result in changes in some target node. 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). Instead, in this project, we chose to perform one-way sensitivity analyses from any node in a network to the ultimate outcome node, using Hugin's Parameter Sensitivity Wizard. This tool allows us to calculate the maximal sensitivity value of each node with respect to the ultimate outcome node.

   A sensitivity value is the first derivative of the sensitivity function, at the point of the probability value. It represents how much the sensitivity function changes at the point. 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 - this means 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. What counts as a large sensitivity value I arbitrarily decide to be > 0.5.

5. Evidence updates the posterior in the correct direction.

   Given a plausible set of evidence (for instance, the set of evidence that a lawyer might present), that we update in chronological order, does the value of the posterior of our ultimate output node change in a meaningful way (corresponding to the evidence strength with regards to the output node)?

3.4.3 Human Criteria

1. How robust is the network against a loss of precision? (Oniśko and Druzdzel, 2013).

   We generated only one Bayesian Network, with the data we collected from the simulation. Then, we create many new networks, with this network as a basis. We do not change the number of nodes, nor the structure of the Bayesian network. We only change the values in each node's cpt. In this case, we round the values in the cpt's, such that the BN becomes increasingly less precise. The intuition behind this is, that when expert users are going to elicit the probabilities, we do not know how robust

the network is against smaller and larger imprecisions in the elicited probabilities. By simulating such imprecisions, we can compare the accuracy and rmsq error of the more imprecise networks to the ground truth of the 'real' network.

We round every value in the cpts c to each of

$$i, i \in \{\text{no disturbance}, 0.01, 0.05, 0.1, 0.125, 0.2, 0.25, 0.33, 0.5\}$$

according to

$$floor(\frac{c}{i} + 0.5) \cdot i.$$

We also add some imprecision into these networks - so that we never have the values of 0 and 1 in our cpts, as this blocks the propagation of evidence throughout the network. Instead, where we would round to 0 or 1, we round to $1 = \epsilon$ and ϵ, where $\epsilon = 0.01$.

2. Could a human find these probabilities?

3. Can a human determine the correct independence relations?

Now that we've established how the automatically generated Bayesian Networks are going to be judged, we can show cautiously in the next chapter how well they are holding up!

Chapter 4

A Simple Stabbing - Non-Spatial Simulations and their Bayesian Networks

4.1 Introduction

This first experiment is not a true agent model. Agents have, as their name suggest, a limited amount of agency. They have knowledge and can interact with the world and each other. This simulation does not contain agents and is instead based on simple rule-based inference.

This chapter is meant to show that we can:

- create a simulation that runs on an uncertain rule-based inference engine.
- create reporters that capture the state of the simulation.
- the output of the reporters can be used to automatically generate Bayesian Networks.
- we can apply our performance criteria to these Bayesian Networks.

Apart from these goals, we will then also compare the networks to the networks created in Vlek et al. (2015). If we find that our method functions well, then we have a chance to apply it to complex spatial simulations in the following chapters. If we cannot make BNs out of this simple simulation, then the rest of our endeavour will be fruitless.

4.2 Method

We want to see if our method can produce similar Bayesian Networks as those in Vlek et al. (2015). This means that we take the two scenarios presented in that paper as inspiration. The facts that both scenarios agree on, are these: Jane and Mark had a fight, and Jane had a knife. Something Happened, and then Mark died.

The problem here, is that Something Happened. The two scenarios that explain why Mark died, are

Scenario 1, Jane stabbed Mark, and then he died.

Scenario 2, Jane threatened Mark with the knife, Mark hit Jane, Jane dropped the knife, Mark fell on the knife, and Mark died by accident.

These two scenarios are represented by two different Bayesian networks (Figure 4.1b and Figure 4.2b for replications of the Bayesian Network structure for scenario 1 and 2 respectively). In Vlek et al. (2015), no probabilities are given, as this paper considers how the structure of the networks represents completeness, consistency and plausibility, and is not meant as an investigation into probability elicitation.

For this experiment, every fact [1] that is represented in the network, is represented as a fact in an uncertain forward chaining inference engine. We are creating a simulation using this engine (this is our data-generating environment). We are creating 3 different simulations: one for scenario 1, another for scenario 2, and a third for a combination, where either scenario 1 happens, or scenario 2, but we do not know which. The output of these simulations is presented to the K2 algorithm, which builds us Bayesian Networks that we can then evaluate.

The uncertain forward chaining engine is the simulation. Every fact has a prior probability, and every conclusion of a rule has a probability given the state of the premises. At every step, the simulation checks which new sentences are true, applies a rule with a given probability, and counts the outcome using the reporters of the same name with the process outlined in the previous chapter.

4.2.1 Behavioural rules for the simulation

This was done using forward chaining, with a time index, which means that a rule could only be triggered at one time (otherwise the probabilities get messed up). So we have a forward chaining rule, which means that if the premises of some rule are true, there are no excluding facts true, the timestep is correct, and the random number generator generated a number that is lower than the probability threshold, we find that the conclusion is true, and add it to our found facts, until we cannot generate any new facts anymore.

[1] Or event, or random variable, or reporter, these are all interchangible in this chapter.

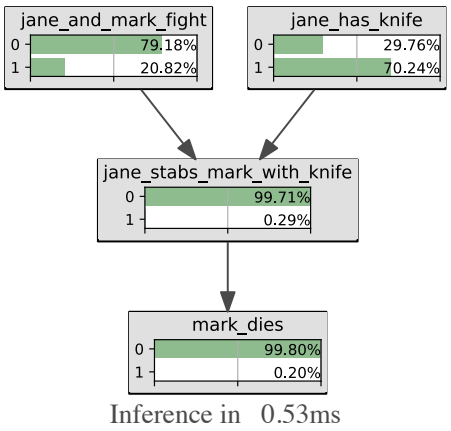
Premise	Conclusion	P(conclusion) given premises
	Jane and Mark fight	20
	Jane has knife	70
Jane and Mark fight, Jane has a knife	Jane stabs Mark with knife	1
Jane and Mark fight, Jane has a knife	Jane threatens Mark with knife	3
Jane threatens Mark with knife	Mark hits Jane	90
Mark hits Jane	Jane drops knife	50
Jane drops knife	Mark falls on knife	10
Mark falls on knife	Mark dies by accident	60
Jane stabs Mark with knife	Mark dies	70
Mark dies by accident	Mark dies	100

Table 4.1: For combined scenarios: blue rules belong to scenario 1, red rules belong to scenario 2, and black rules belong to both.

The probabilities for these rules to fire were chosen as by the modeller as a subjective estimation of how often these events 'might happen', and do not correspond to real statistics. Each simulation was run 50,000 times.

4.3 Results

We succeeded in generating three different BNs, one per scenario (Figure 4.1a, Figure 4.2a, Figure 4.3, with the networks from Vlek et al. (2015) on the side for comparison).



(a) Automatically generated.

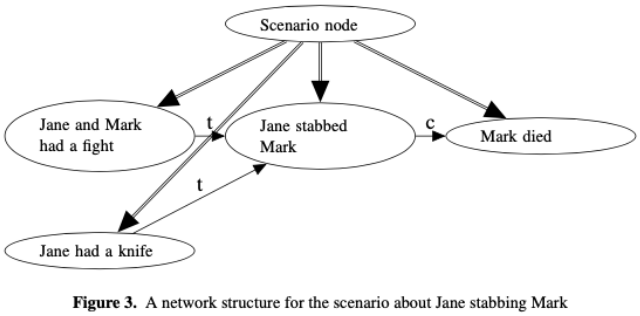
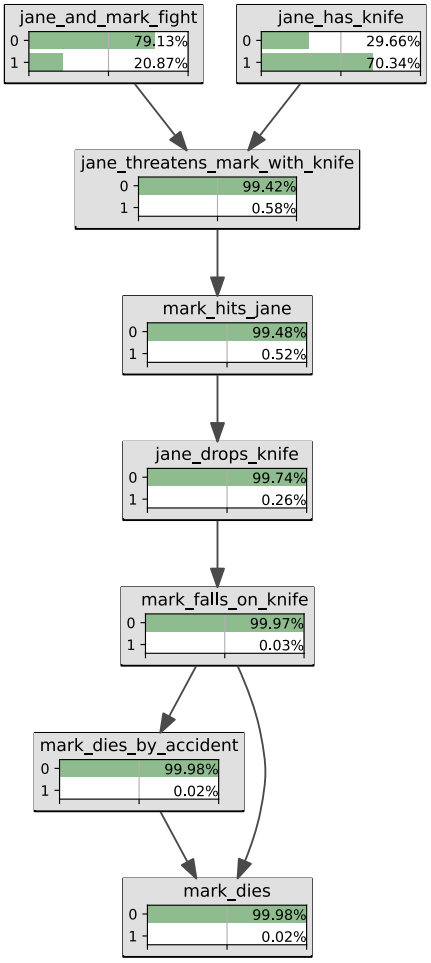


Figure 3. A network structure for the scenario about Jane stabbing Mark

(b) Original BN.

Figure 4.1: Scenario 1



(a) Automatic generated.

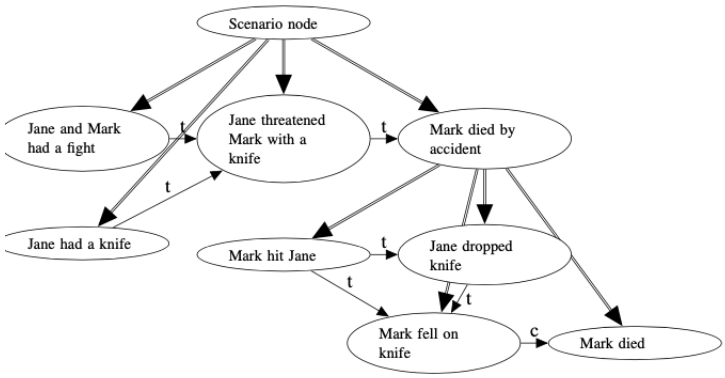


Figure 4. A Bayesian network structure for the second scenario

(b) Original BN.

Figure 4.2: Scenario 2

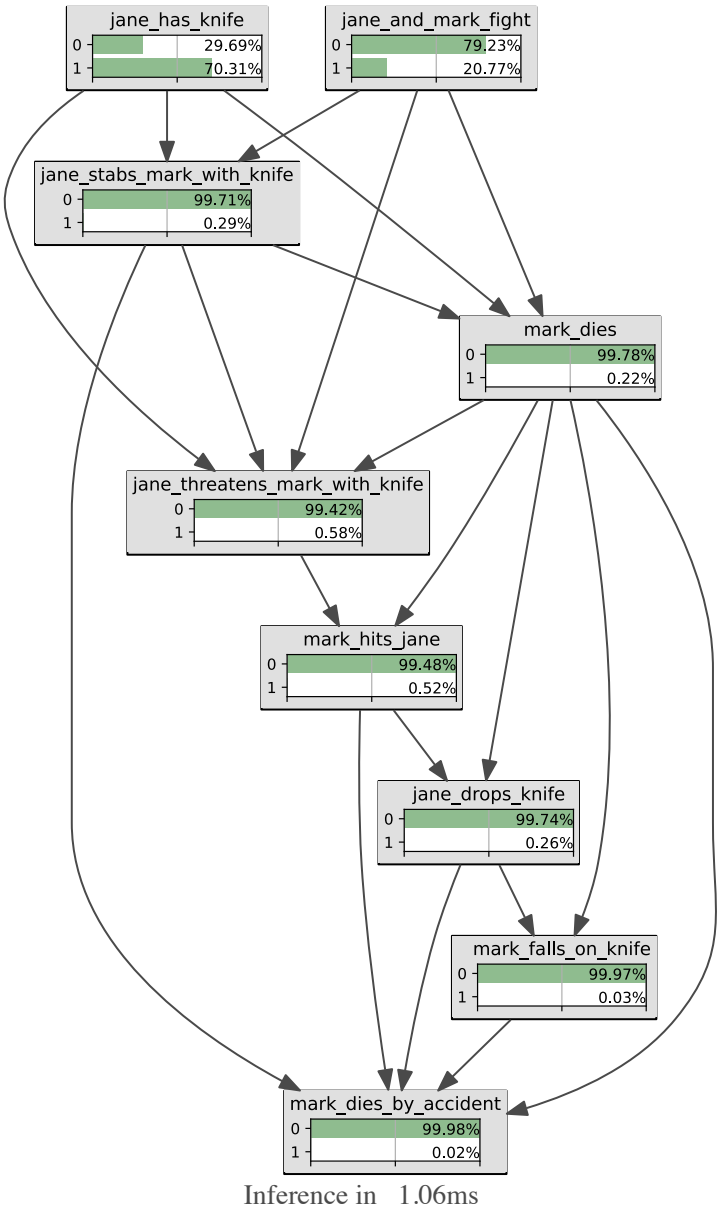


Figure 4.3: Automatically generated BN with rules from both scenarios included.

4.3.1 Evaluation with regards to structural, performance and human-factor criteria

In this section, the evaluation criteria as set out before are applied to the networks.

Structural Criteria

1. **Hypotheses are ordered temporally.**

   We see that in both the separate networks, the temporal ordering of the nodes is correct - every node has as a parent a node that represents an event that would have happened earlier. In the combined network, this is not the case: there are some temporal links - the chain of events from "Jane threatening Mark" to "Mark dies by accident" is represented correctly, however, every node in this chain has as a parent "Mark dies", which would, temporally, occur only after "Mark falls on knife" occurs.

   The combined Bayesian Network brings up difficulties for temporal ordering. If we try to merge two mutually exclusive scenarios, we cannot meaningfully speak of that 'Jane stabs Mark' happens before or after 'Jane threatens Mark' - either one of these events happen, never both, and no one of them will happen before or after the other. This is not a problem for the performance of the Bayesian Network - we can arbitrarily decide on a parent node, and interpret the arcs as conditional, and not temporal or causal. However, structurally, it is easier to think about the links as temporal, and temporally organised 'within' scenarios.

2. **Evidence connects to hypotheses.**

   In these networks and simulation, no evidence is implemented yet, that means that this criteria does not apply.

3. **Relevance: All relevant events are in the BN, all irrelevant events are outside of the BN.** This criteria, and the next one, are freebies, because we are just copying a network that already exist - our set of relevant events is the same as that in the original network by Vlek. By the structure of the network, we see that all events are connected to each other, there is no node that is both without parents and without children, which means that all nodes are relevant in that they represent conditional relations.

4. **Independent events are not connected to each other.**

   This is done correctly in the non-combined networks, where each node has appropriate parents. However, in the combined network, we see that this goes wrong.

Performance Criteria

We consider the accuracy and root mean square per network, then also look at the correspondence, sensitivity values and the progression given a certain evidence set.

1. **Accuracy and Root Mean Square error**

   The accuracy and rmse are shown in Table 4.2.

Network	Accuracy	Root Mean Square
Scenario 1	0.83	0.17
Scenario 2	0.88	0.14
Combined	0.81	0.2

Table 4.2: Accuracy and Root Mean Square Error for the networks

2. **Correspondence.**

   In this case, we know exactly the frequencies of events given their premises in the rule base, because we have set them explicitly in our probabilistic-forward-chaining engine. These frequencies are represented in Table 4.5 as the 'True P(event)' column. The 'P(event)' column is the result of the prediction of the Bayesian Network. As shown in the tables, the Bayesian Network **usually** estimates the probability of the event within ±1% uncertainty. There are a few exceptions to the general behaviour of less than ±1% uncertainty in the network - the network generally overestimates small probabilities. Whether an uncertainty of 'generally less than ±1%' is good or bad depends on the precision that the network requires - since we do not have any evidence that really depends on high-precision data or networks (eg: a change of 0.01 in one node results in a different conclusion - see sensitivity analysis and precision reduction), the Bayesian Networks seems satisfactory.

3. **Sensitivity analysis** (Figure 4.4a, 4.4b, 4.4c) shows that no irrelevant event influences some output, as well as being robust, as the largest sensitivity values remain smaller than 1. The sensitivity values for the second scenario in the full network are almost insignificantly small, showing that they have very little influence over the 'mark_dies' node.

   From these values we can see that a parameter chance in most nodes will not affect the output probability very much. The parent nodes of the output node are, of course, the most influential.

4. **Effect of evidence on the posterior.**

   We plot the effect of a sets of evidence on the posterior (Figure 4.6b, 4.5b, 4.5c). In each, we select a different story:

Conclusion	True P(event)	P(event)
Jane and Mark fight	20	20.87
Jane has knife	70	70.23
Jane stabs Mark with knife	1	2.05
Mark dies	70	75.46

Table 4.3: For only scenario 1.

Conclusion	True P(event)	P(event)
Jane and Mark fight	20	20.87
Jane has knife	70	70.23
Jane threatens Mark with knife	3	3.88
Mark hits Jane	90	90.21
Jane drops knife	50	50.09
Mark falls on knife	10	11.13
Mark dies by accident	60	61.25
Mark dies	100	99.79

Table 4.4: For only scenario 2

Conclusion	True P(event)	P(event)
Jane and Mark fight	20	20.87
Jane has knife	70	70.17
Jane stabs Mark with knife	1	2.06
Jane threatens Mark with knife	3	3.88
Mark hits Jane	90	91.50
Jane drops knife	50	52.97
Mark falls on knife	10	10.92
Mark dies by accident	60	66.33
Mark dies (premise: Jane stabs Mark)	70	68.68
Mark dies (premise: Mark dies by accident)	100	98.77

Table 4.5: For combined scenarios

Node	Max
jane_and_mark_fight	9.83E-3
jane_has_knife	2.91E-3
jane_stabs_mark_with...	0.39
mark_dies	1

(a) Sensitivity values for scenario 1 network.

Node	Max
jane_and_mark_fight	9.83E-3
jane_has_knife	2.91E-3
jane_stabs_mark_with...	0.39
mark_dies	1

(b) Sensitivity values for scenario 2 network.

Node	Max
jane_and_mark_fight	0.01
jane_has_knife	3.07E-3
jane_stabs_mark_with...	0.1
mark_dies	0.56
jane_threatens_mark_...	0
mark_hits_jane	0
jane_drops_knife	0
mark_falls_on_knife	0
mark_dies_by_accident	0

(c) Sensitivity values for combined network.

Figure 4.4: Sensitivity Analysis

Jane and Mark fight	1	1	0	0
Jane has knife	1	0	1	0
P(jane_stabs_mark_with_knife)	0	0	0	0

Table 4.6: CPT of jane_stabs_mark_with_knife

For the first case, Jane has a knife, and Jane and Mark have a fight, but then Jane does not stab Mark - hence, we see the posterior of 'mark dies' increase once we learn that they have a fight, and decrease when Jane does not stab Mark.

For the second case, Jane and Mark fight, and Jane has a knife, and Jane threatens Mark with a knife, Mark responds by hitting Jane, and Jane drops the knife. All of these facts cause the posterior of 'mark dies' to increase, since, given all of these factors, we would expect it a bit more likely that Mark dies. However, it is a very small increase, from 0.0 to perhaps 0.1. Then, once we learn that Mark actually does fall on a knife, we see the posterior probability of Mark dying increase by much more, growing to 0.5. However, Mark gets lucky, and does not die accidentally (he probably fell on the good side of the knife). Now, the posterior for 'mark dies' also goes to 0, since the only way that Mark dies in this scenario is due to an accidental death by falling on a knife.

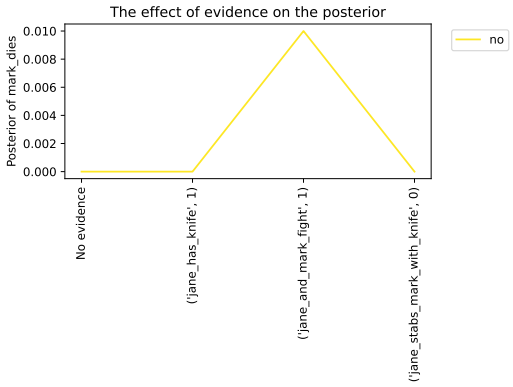
The combined scenario represents the story where Jane and Mark fight, Jane has a knife, but Jane does not stab, but threatens instead. Then Mark hits Jane, and Jane drops the knife, which is the highest posterior probability for 'mark dies' in the entire story. Then, Mark does not fall on the knife, and Mark does not die by accident, and hence the posterior for 'mark dies' goes to 0.

Human Criteria

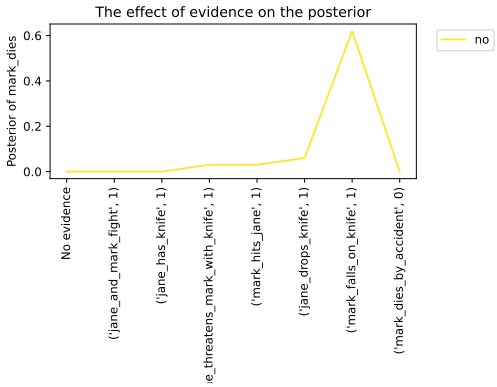
1. **How robust is the network against a loss of precision?**

   We tested this by changing the cpts to be less and less precise, and plotted the results in Figure 4.6a - 4.6f. We can see that the results are very drastic. At around a loss of precision of 0.05 (less for the combined network), the accuracy of the predictions drops from around 80% to 0%.

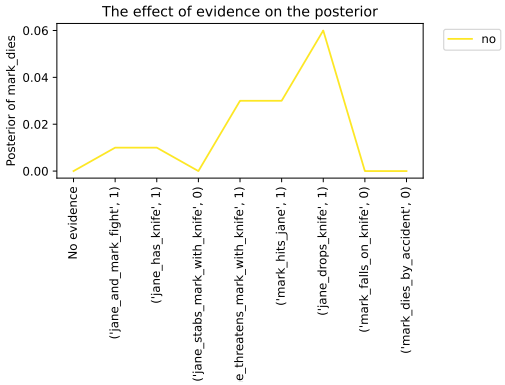
   This dramatic is altogether due to entering inconsistent evidence inside the networks. For example, in the rounded to 0.05 network of the first scenario, trying to add 'jane_stabs_mark_with_knife' as being true, results in the network failing. This is not due to the 'mark_dies' node, but instead due to rounding away to 0 and 1 in the cpt table for 'jane_stabs_mark_with_knife' (Table 4.6). Entering the evidence that Jane did stab Mark is inadmissible according to the CPT, hence the failed state.



(a) network 1.



(b) network 2.



(c) Combined network.

Figure 4.5: Posteriors

But what if we do not round to 0 and 1, but round to ϵ and $1 - \epsilon$. Here ϵ would be an approximation of 0 and 1, just purely meant to 'unblock' the inference in the Bayesian Networks: going from 'never possible' to 'not impossible'. If it worked, it would be a pragmatic solution. Hence, the rounding algorithm was adapted to use $\epsilon = 0.01$ - all values that were previously rounded to 1 are instead now 0.99 and all 0's become 0.01.

This intervention is successful. Figure 4.7 shows the accuracy of networks under lower and lower degrees of precision, but now rounding to ϵ. We see that after an initial small drop in performance in scenario 2 and the combined scenario, the accuracy of the adjusted networks remains constant at around 80% in all cases. The only effect we see of the lowered precision is the fact that the RMSE increases slightly as the precision lowers.

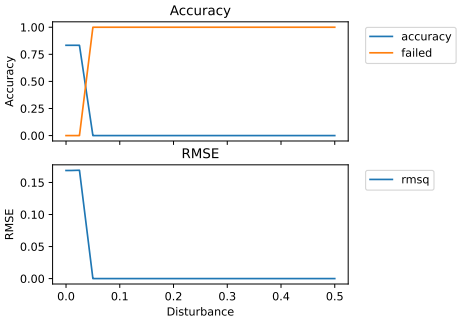
2. **Could a human find these probabilities?**

Using the ϵ-networks with lower precision, like rounding to 0.1, 0.25 or even 0.33, is a promising approach. Especially in cases where a node has multiple parents, being able to fill out probabilities to a lower precision is very useful. This low-precision approach also lends itself well to conversions to a verbal probability scale (Renooij and Witteman, 1999).

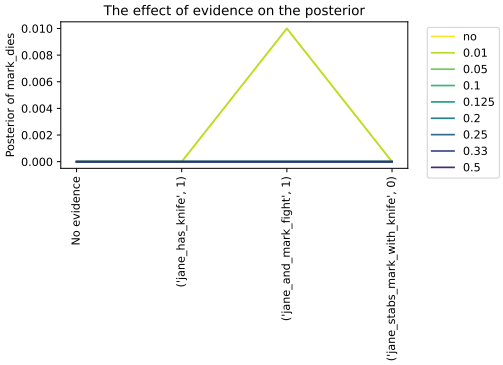
However, when you round the cpts to a lower precision, you are still rounding based on the original value. If you did not have the original value, would you be able to think of it? The original network has as prior probabilities for the $P(fight) = 0.2$, and $P(knife) = 0.7$. It is unclear how these values would be elicited from experts, if this were a real court case.

For one, how to operationalise these variables? We have it easy in our simulation 'Jane has a knife' is true if and only if the reporter for 'jane has a knife' is true, which is based on a random number generator. However, in real life, what do we mean? Do we mean all situations where Jane is in possession of a knife? A lot of these situations are not relevant, if Jane is an adult woman, she probably owns a knife in her kitchen, and we would say $P(jane\_has\_knife) = 1 - \epsilon$. If instead we mean that we consider all the situations where Jane has a knife within reach, we know that that that probability is going to be lower than $1 - \epsilon$ - but out of all the possible 'situations' that Jane can be in, we would consider that probability to be ϵ: if we consider all seconds of your life and consider the proportion of those where you had a knife within reach, that would be a proportion smaller than $\epsilon = 0.01$. This is also not useful.

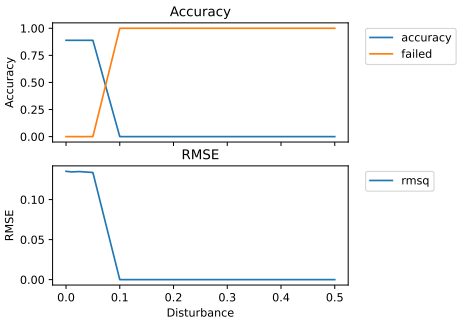
If we cannot determine what reference class to take for 'Jane has a knife', we might just give up and assign the node a probability of 0.5. This is what we do in the lowest-possible precision level, rounding everything to $\{\epsilon, 0.5, 1 - \epsilon\}$ - the resulting



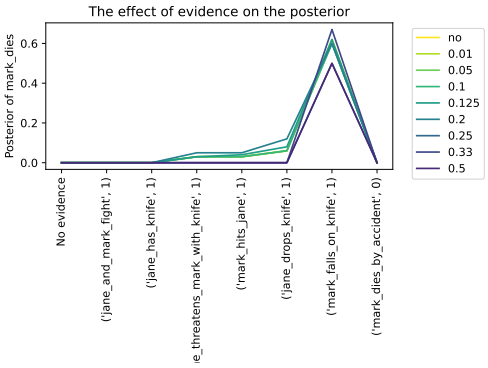
(a) Network 1 accuracy and RMS.



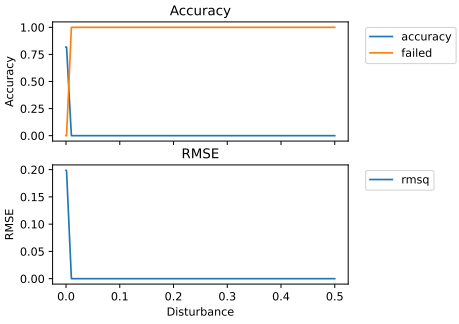
(b) Network 1 posterior.



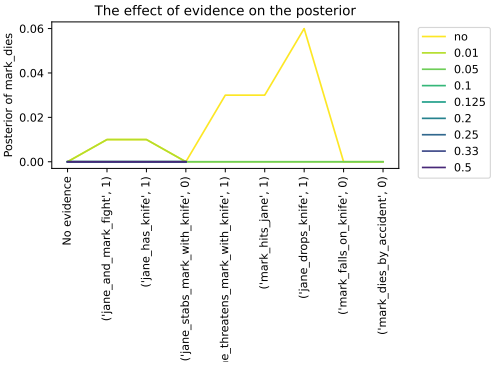
(c) Network 2 accuracy and RMS.



(d) Network 2 posterior.

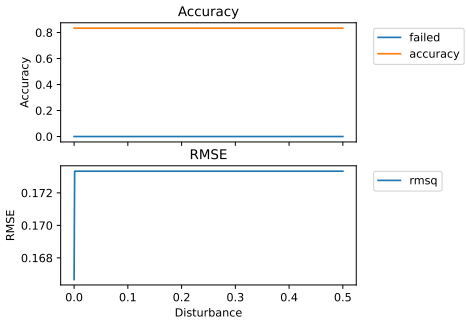


(e) Combined network accuracy and RMS.

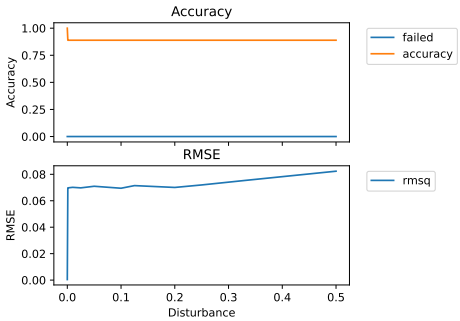


(f) Combined network posterior.

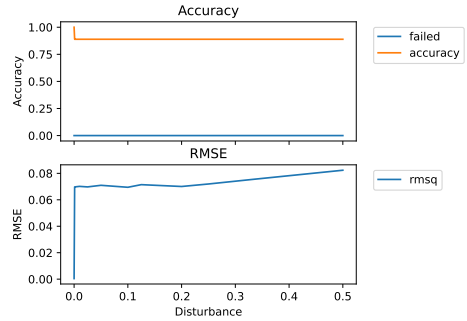
Figure 4.6: Performance of networks



(a) Network 1 accuracy and RMS.



(b) Network 2 accuracy and RMS.



(c) Combined network accuracy and RMS.

Figure 4.7: Using error terms instead of 1 and 0 in the networks improves performance immensely.

network still has a high accuracy. So in the case of these networks, it might actually be plausible to use Bayesian uncertainty priors.

3. **Can a human determine the correct independence relations?** These rounded networks are in-part accurate, because they are using the same structure as the original network. We do not have to derive the structure independently, based on no information. We already have the structural information based on the information from the simulation. Would you be able to derive this structure without using the simulation, or only while having limited information?

   In case of scenario 1 and 2, certainly a person would be able to determine the correct independence relations: these scenario is very temporally ordered, and we see clear cause and effect. An additional relation between 'jane_and_mark_fight' and 'jane_has_knife' might be appropriate, since, thinking naively, Jane might be more likely to have a knife if she knows that she and Mark have a history of fighting and/or violence. However, this is not modelled in the original scenario (Vlek et al., 2015), and hence not modelled in the simulation either.

   The combined scenario, however, is more complicated. While there is some temporal organisation, since the parents of nodes usually occur before the nodes themselves, there are more arcs between nodes than in either of the two independent scenarios. The temporal organisation is also broken by the nodes 'mark_dies', which might happen after Mark threatening Jane, but is the parent of that node nonetheless (and the rest of that chain of events). A second atemporality occurs between the stabbing and threatening node (see discussion in Comparison below).

   Dependencies between Mark hitting Jane and Mark dying by accident should be irrelevant based on the events that could happen in between (like Jane dropping the knife, and Mark falling on the knife). Looking at the cpt of 'mark_dies_accidentally', there are a lot of replicated values. It is unclear how the algorithm should be adapted to reduce the amount of unnecessary dependencies.

4.4 Discussion

In the discussion, we focus on the comparison of the automatically generated networks to the paper in Vlek, and consider some other issues.

4.4.1 Comparison to Vlek's Method

The Bayesian Networks generated using the method outlined in this paper diverge meaningfully from the networks as presented in Vlek et al. (2015).

Firstly, in the original paper there was no integration of both scenarios into one network.

In this paper, the combination is shown in Figure 4.3. For the integration of two scenarios, Vlek proposed combining the networks from Figure 4.1a and Figure 4.2a using the merged scenario idiom construction as outlined in Vlek et al. (2014). This merging depends on a scenario-like construction with a constraint node, and requires extra probabilities to be elicited. In this case, there was no 'merging' of two networks, instead the simulation was run with all the rules and reporters (and one excluding rule that states that if Jane was threatening Mark, Jane couldn't stab Mark). In short, the 'merging' happened at the simulation level, and the data obtained from the simulation was then used to create the network that combines both scenarios (Figure 4.3), effectively effortlessly combining two networks.

Secondly, the network structure between the networks in Vlek et al. (2015) and this paper are different. In this case, there is no scenario node or subscenario-structure organisation. The only organisation in the automatically generated networks is the 'temporal' structure of parent vs child nodes, which is usually present (the child event happens after the parent event). The subscenario-structure we see in Figure 4.2b is not present. The lack of scenario, subscenario and constraint nodes, is because these are not necessary in representing the events of the simulation. A 'scenario' is not ordered using a scenario node, but instead should be modelled in such a way that when setting evidence, some nodes being true 'cause' other nodes to become false (and vice versa). The scenarios, in this project, emerge from the network, instead of being represented explicitly.

The temporal structure is not consistently present in the networks, however. In Figure 4.3, we see that the node 'mark_dies' is the parent node of almost all the nodes of the second scenario. This is because the temporal ordering in K2 is not straightforward when multiple scenarios are combined (eg, Mark dies after being stabbed, which is the 4th and final event in scenario 1, but Mark can also die after falling on the knife, which is the 6th event in scenario 2). Either improving the temporal ordering algorithm or the ordering of nodes by the modeller could improve this situation.

Even though the networks in this paper are different from the networks in Vlek et al. (2015), we can still use the criteria of completeness, consistency and plausibility outlined in that work to assess their quality.

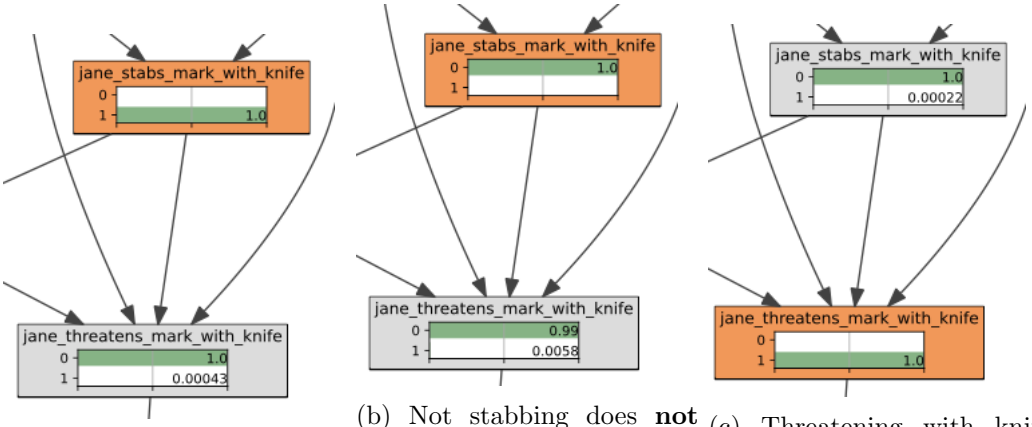
- Completeness can be operationalised in this situation, not by looking at story schemes and idioms, but by looking at the reporters in the simulation. A network can be considered complete if it represents every reporter as a node. This means that all the networks are complete, because every network contains nodes that correspond to all reporters. It is more likely that we suffer from the opposite problem, where irrelevant reporters are still included in the network.

- Consistency means that there are no internal contradiction. In this network, consistency cannot be seen structurally (by separate scenarios and constraint nodes). The

node ‘jane threatens mark with knife’ and the node ’jane stabs mark with knife’ are inconsistent, but they belong in the same network. Instead of structurally, consistency is ensured using the conditional probability tables. The relations of these nodes are such, that when one is true, the other becomes false (Figure 4.8).

This means that several different scenarios can be combined into one network, while the overall network is still consistent. Both scenarios share some events (as scenarios can do in real life), but the different scenarios themselves are not separated from each other by structure, and we do not need the constraint node as suggested in Fenton et al. (2011). Fenton suggests two objections against a solution as presented here: unnecessary extra numbers, and lack of causal interpretation. Unnecessary extra numbers is not a problem here, because we do not need to elicit them (they fall out of the algorithm automatically). For causality, we are not interpreting the arcs between nodes causally in any way. They are conditional, and in that case it does not matter which node is the parent.

Other types of inconsistency - such as Jane stabbing Mark, when Jane does not have a knife, result in a ‘conflict’ message when when the network is imported into Hugin. PyAgrum does not show such a conflict message.



(a) Stabbing means no threatening.

(b) Not stabbing does **not** mean that there’s threatening, we do not have mutual exclusivity constraint.

(c) Threatening with knife means that there’s no stabbing.

Figure 4.8: Effect of setting evidence on contradictory nodes.

- Plausible. Plausibility is defined such that a scenario should correspond to the modellers knowledge about the world. In this case, we have outsourced the problem of plausibility to the simulation, and the Bayesian Network reflects it.

  If we set to true one of the nodes in the temporal chain that leads towards the

outcome of scenario 2, then we find that the rest of the nodes in that chain become more plausible - but not by much, because scenario 2 is very implausible to begin with! For example, P('mark_falls_on_knife' $|\emptyset$) = 0.00028, but once we know that Jane has threatened Mark, we see P('mark_falls_on_knife' | 'jane_threatens_mark_with_knife) = 0.046. Implausible elements become more plausible once we find support for them, reflecting the modeller's knowledge about the world.

So our overall conclusion with regards to the quality of the network, is that the networks represent the scenarios well with regards to completeness, consistency and plausibility, even without structural support from scenario, sub-scenario and constraint nodes. This means that this method seems promising and does not need to include or build on the scenario-type structures.

4.4.2 Further Discussion

Are we just outsourcing?

This approach to completeness and plausibility outsources the problem of determining completeness or eliciting plausibility from the builder of the network to the builder of the simulation. We get a complete network for free, given a simulation, in our approach, but that is because we are basing our network entirely on the simulation. This brings up the following problem: what does it mean for a simulation to be 'incomplete'? How can we know that the probabilities inputted in the simulation are plausible?

There is no good answer to this question. A simulation will always be an incomplete reflection of reality. Perhaps the simulation should correspond to some story or idiom. If the scenario contains 'intentional' agents, we might find it easier to think about stories and idioms in simulation than we find it to think about stories and idioms with regards to Bayesian Networks. However, this needs to be investigated further.

Bayesian Networks as a guide to reasoning

Ideally, we would want a BN to protest when you try to add evidence to it that corresponds to an impossible world state. This should cause conflict between nodes, or at least result in a low accuracy. We calculated accuracy over the set of all possible world states, however, if we take the accuracy over the set of all impossible world states (eg: states that never occur in the simulation), we would want to see a low accuracy. So, the accuracy over the impossible world states is shown in Table 4.7 , shows for the generated ϵ-networks.

Here, the accuracy is remarkably low, and the RMSE is high. This means that if you, by accident, fill in an impossible world state as evidence in your BN, the probability that you predict the output node correctly becomes very low. However, we also see that these ϵ-networks do not break - the number of times that we find an inconsistency error is 0

Network	Accuracy	Root Mean Square	Inconsistency error
Scenario 1	0.3	0.70	0
Scenario 2	0.481	0.516	0
Combined	0.491	0.509	0

Table 4.7: Accuracy and Root Mean Square Error for the networks calculated on the impossible states.

for all networks. In the best case scenario, we want the network to break if you fill in inconsistent evidence, and not just produce a low accuracy (since you can only calculate accuracy over a data set).

Resolution and number of runs

The accuracy of the Bayesian Network improves as the number of runs improves. This is an obvious fact. It has implications for the rest of the network. For example, if we run the network too few times (let's say 10 times), then the network will not be able to estimate the probability of certain states - eg: the probability that Mark hits Jane can be calculated as: $0.7 \cdot 0.2 \cdot 0.03 \cdot 0.9 = 0.00379$, which means that there's a 0.378% chance that it happens, eg, if we run the simulation a 1000 times, Mark will hit Jane in 4 of them (or, actually, the sentence "Mark hits Jane" is true in 4 of them). If we go all the way down the chain of unreasonable facts to "Mark dies accidentally", we get a probability of 0.0001134, which is 1 every 10.000 runs. If we run the simulation 10.000 times, that means that we cannot estimate probabilities that are smaller than 0.0001 - either they occur 'accidentally', eg, by chance, and the network estimates that they happen once every 10.000 runs, or they don't happen within 10.000 runs, and the network estimates that these events never happen at all - when in fact, they might happen, but just once every 20.000 runs.

What are the practical implications of this? On this side, that we need enough data to get an accurate prediction. But, if we're thinking about elicitation, this means that if you decide to add an event to a scenario, or a node to a network, you really have to think about the probability that you assign it - a probability of 1 in 10.000 would really mean that this event would only happen once every 10.000 situations.

Future Research

This simulation and network are actually not a good representation of agent-based modelling. There are no agents acting with intentions or beliefs or knowledge, it's just a global forward-chaining engine firing off a rule when a random number generator says that it can. There was no emergence of probabilities due to interactions of agents with their environment, or with each other. This type of emergent behaviour and intention is essential for modelling crime cases, and will be introduced in the next chapter.

There were also no separate evidence nodes, because the original networks and scenarios did not contain evidence. Evidence will also be introduced in the next chapter.

4.4.3 Conclusion

Even this very simple 'forward chaining' simulation (forward chaining with some probabilities attached), can be used to construct meaningful Bayesian Networks. The generated Bayesian Networks have generally high accuracy and low RMS error, even when their cpts are disturbed, when we use an error term instead of 0. The Bayesian Networks reflect the structure of the simulation.

This shows the success of the pipeline. We can make a simulation using forward chaining, that generates world-states, and those world states can be translated into Bayesian Networks by means of the K2 algorithm, with reasonable accuracy even under decreased precision. The K2 manages to replicate constraints and merge two different possible scenarios together. The resulting networks are consistent, complete and plausible. The scenario, sub-scenario and constraint node mechanisms are not necessary for modelling these Bayesian Networks.

So if the real world worked like a simple forward chaining inference method where we knew exactly when a proposition was true (which is, if the state is true, we set the proposition to true in every epoch), then BNs would work flawlessly, even if we weren't very great at estimating the correct frequencies precisely.

Chapter 5

A Simple Robbery - Spatial Simulations and their Bayesian Networks

The problem is that in the previous chapter, we have established that we have a method to convert a simple forward-chaining simulation into a collection of data, which then in turn can be used to create a Bayesian Network, that represents the situation in the simulation.

However, this was totally 100% determined by deterministic forward chaining rules. The real world does not run on deterministic forward chaining rules (as far as we know). At least, we are spatially situated, which means that there are probabilities that will arise out of interactions between agents and their environment. These probabilities are not 'set' in the same ways as the probabilities are set in the previous chapter, they arise organically from interactions: an agent has to be located at a door to break in, an agent can be seen by a camera only in some locations of the simulation. We do not know these probabilities a-priori.

Research questions for this chapter:

- Can we create an automatic BN from a simulation of similar accuracy and rmse as in the previous chapter, but now spatially explicit, with agents, and with evidence?
- What is the effect of loss of precision on this network?
- What is the effect of truly private knowledge on the network?

5.1 Experiment 1: Generating BNs and lowering precision

For the first experiment, a very simple crime case will be modelled in a multi-agent simulation. This is the data-generating environment, the gathered data will be used by the K2 algorithm to automatically generate a Bayesian Network. This network will be assessed based on the criteria set out in chapter 3.

5.1.1 Introduction

The scenario that the agents play out goes as follows. There are two agents. They are neighbors. One day, the agent who lives in the house on the right (the thief), walks past the house of the other agent (the victim) [1]. The thief looks inside, if it can. Sometimes it cannot look inside, because the victim has drawn the curtains. However, if the thief can look inside, it might see a goodie in the house. It also might not see a goodie, if the victim has lost the goodie - in that case, the goodie has been lost. However, if the goodie is there, the thief will know that the goodie is there. Then, because the thief is a rational agent, it will try to calculate the worth of the object, and if it might want to steal it. It can decide to target the object. If the thief knows and targets the object, the thief has a motive. Then, the thief attempts to break into the house by breaking the lock on the door, grabbing the goodie, and running away to its own house. The thief has a breaking-and-entering skill (random, 0, 10), meaning that it can fair to break the lock, and also a detail-oriented skill (random, 0, 10), meaning that it can disturb the house while robbing (leaving evidence). Meanwhile, the victim is returning home. If the thief thinks that the victim might have seen it stealing, the thief will flee home, with or without the goodie. The victim has also hung up security cameras on the outside corner of its house, that might serve as evidence against the thief.

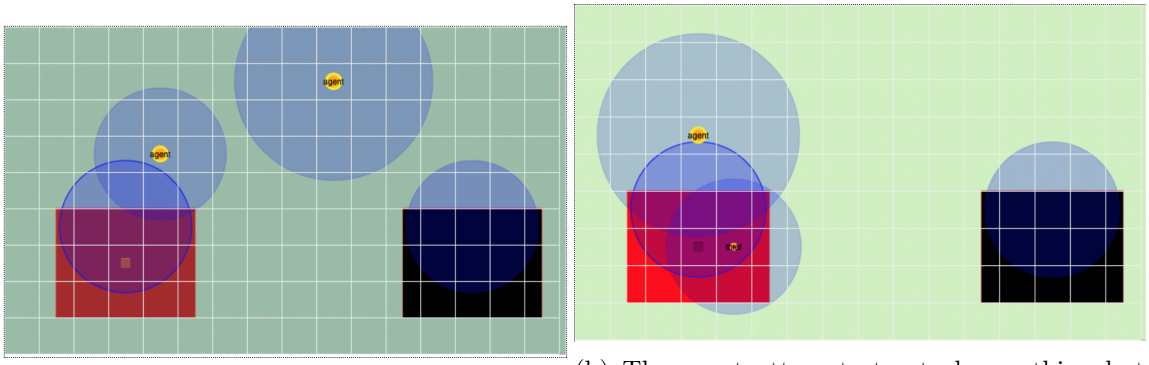
With regards to the end, there can be three outcomes: either the goodie is lost, the goodie is not lost but was stolen successfully, or the goodie is not lost and wasn't stolen successfully. The environment of the agents is very simple, just two 'houses' (rectangles, really). The environment and the agents (with their vision parameter) are shown in Figure 5.1.

5.1.2 Methods

We created a simulation of the situation described in the introduction, using the MESA multi-agent framework in Python. There are two agents, who are walking around in a discrete grid. Every simulation was simulated for 30 epochs, because all the possible scenarios could be described within 30 epochs. We performed 2000 iterations.

The behaviour of the agents is described in Figure 5.2.

[1] It is not legally correct to talk about thief and victim at this point in the simulation, because nothing has been stolen yet. However, it's convenient!



(a) The simulation with the 2 agents.

(b) The agent attempts to steal something but flees when it thinks that it is noticed.

Figure 5.1: Agents behaving badly in simulation.

The actions of the agents correspond to reporters. The operationalisation of the reporters (random variables) are described below:

lost_object a random number generator (0, 1) generates a number. If the number ≤ 0.2, the object is lost.

curtains a random number generator (0, 1) generates a number. If the number ≤ 0.8, the house has no curtains.

raining a random number generator (0, 1) generates a number. If the number ≤ 0.5, it is raining.

know_object if the thief sees an object in our vision that is not our own, and the thief is not already targeting something else.

target_object if the thief know the object exists, and it considers the value of the object higher than its risk threshold - where the range is (random, 0, 1000).

motive if the thief has a target.

compromise_house if we are adjacent to the target's house's door, and we have a breaking and entering skill of greater than 5.

flees_startled if there is another agent in the thief's vision when we have a motive.

successful_stolen if the thief is outside with the object and has not fled.

E_s_spotted_by_house if the thief is within the visual radius of the camera belonging to the victim's house.

E_disturbed_house if the thief is inside the house, and the thief's detail-oriented skill is

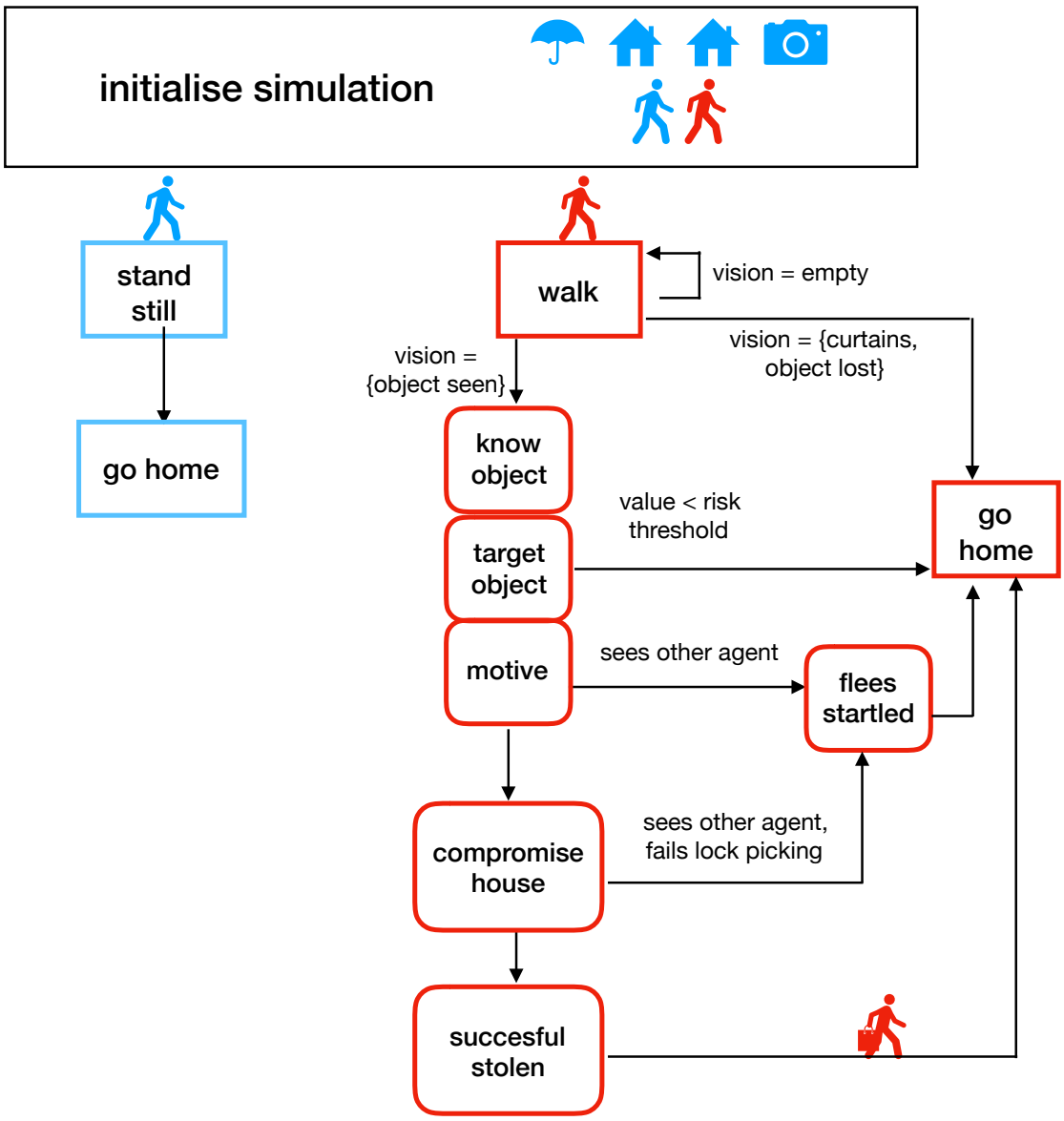


Figure 5.2: The behaviour of the agents. The nodes with rounded edges correspond to the nodes in the Bayesian Network.

smaller than 8.

E_object_is_gone if the object is not in its original location when the victim agent comes home.

E_broken_lock if the house is compromised.

E_s_spotted_with_goodie if the thief is within visual range of the camera and it has the object.

E_private if the thief sees another agent in its visual range.

5.1.3 Results

The generated Bayesian Network is shown in Figure 5.3.

Evaluation with regards to structural, performance and human-factor criteria

In this section, the evaluation criteria as set out before are applied to the networks.

Structural Criteria

1. **Hypotheses are ordered temporally.**

   The main scenario hypothesis nodes are ordered temporally.

2. **Evidence connects to hypotheses.**

   All evidence is connected to at least one hypothesis. However, the structure of how the evidence nodes are added is confusing and overwhelming. One piece of evidence - the most crucial one, which is that the evidence is gone, has 5 parents!

3. **Relevance: All relevant events are in the BN, all irrelevant events are outside of the BN.**

   We know that truly irrelevant nodes are kept out of the main Bayesian Network - we can see that the node for 'raining' is not connected to any other node. This is because the rain does not affect anything in this simulation - this is reflected in the Bayesian Network.

4. **Independent events are not connected to each other.**

   Evidence is connected to each other. This is because we need to consider dependent evidence as dependent in the network (see (Fenton et al., 2012) for explanation of dependent evidence).

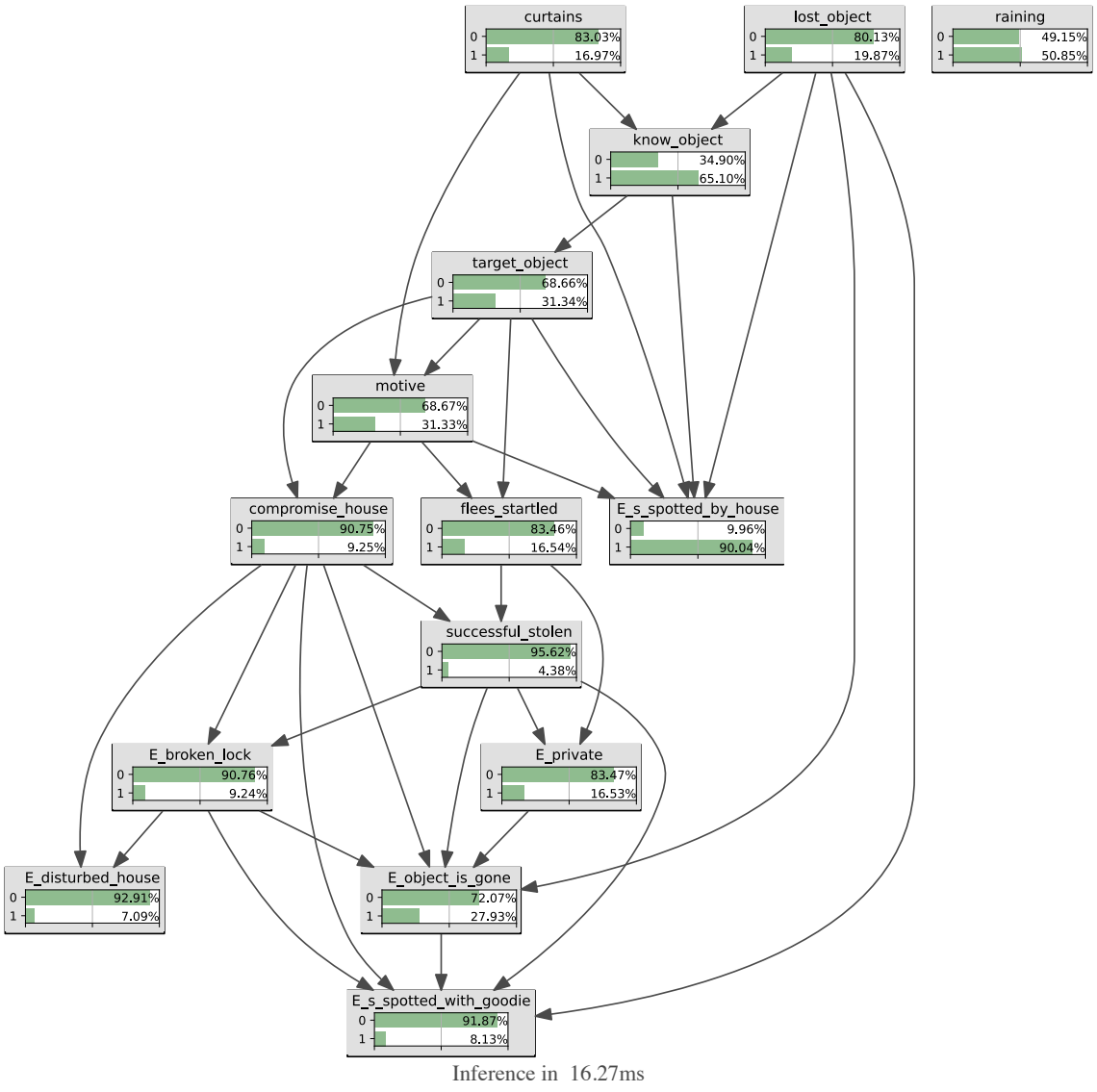


Figure 5.3: network structure

Conclusion	Frequency P(event)	BN P(event)
lost_object	0.1985	0.1987
curtains	0.1695	0.1697
raining	0.5085	0.5085
know_object	0.6515	0.6510
target_object	0.3135	0.3134
motive	0.3135	0.3133
compromise_house	0.0925	0.0925
flees_startled	0.1655	0.1654
successful_stolen	0.05	0.0438
E_s_spotted_by_house	0.901	0.9004
E_disturbed_house	0.071	0.0709
E_object_is_gone	0.281	0.2793
E_broken_lock	0.0925	0.0924
E_s_spotted_with_goodie	0.083	0.0813
E_private	0.1655	0.1653

Table 5.1: Correspondences

Performance Criteria

We consider the accuracy and root mean square per network, then also look at the correspondence, sensitivity values and the progression given a certain evidence set.

1. **Accuracy and Root Mean Square error**

   The accuracy of the network is 87.9% and Root Mean Square error is 0.14. This is comparable to the networks in the previous chapter.

2. **Correspondence.**

   Once again, compared the probabilities in the network with the frequencies of the events in the simulation (maybe run again, larger training set but don't actually train on it). We see the same pattern (Table 5.1) as before, it is accurate within ± 0.005.

3. **Sensitivity analysis values**

   The sensitivity analysis shows (Figure 5.4) that a shift in the parameters for 'compromise_house' has the largest effect on the output node 'successful_stolen', but all the sensitivity values are low, meaning that a small shift in the parameters of the nodes do not result in a greater shift in the ultimate node - hence, the network is robust.

4. **Effect of evidence on the posterior.**

Node	Max
target_object	0.09
successful_stolen	0.79
raining	0
motive	0.04
lost_object	0.05
know_object	0.04
flees_startled	0.09
curtains	0.05
compromise_house	0.69
E_s_spotted_with_goodie	0
E_s_spotted_by_house	0
E_private	0
E_object_is_gone	0
E_disturbed_house	0
E_broken_lock	0

Figure 5.4: The maximum sensitivity values for every node as it relates to 'successful_stolen' as presented in Hugin.

The effect of turning on evidence on the posterior of the initial network is shown in Figure 6.6.

The scenario that this posterior progression corresponds to is as follows: the duped agent comes home, and finds that their object is gone. Then, it takes a closer look at the door, and sees that the lock is broken. It looks at the disturbed house, and then at the cameras. The camera shows that the other agent was spotted near the house, and later on emerged with the object. Then, we need to look inside the head of the stealing agent, who fled before they saw the duped agent come home - hence, E_private is 0.

We see the effect of evidence strength in this progression: once we know that the lock is broken, seeing that the house is disturbed does not add anything to our estimation of the posterior of being robbed, and spotting the suspect near the house also does not change the posterior. Only when we see the suspect with the robbed object, the posterior starts to increase again. This corresponds to intuition.

Human Criteria

1. **How robust is the network against a loss of precision?**

   We again create networks with lower precision, and use the ϵ method to create networks that do not saturate with 1 or 0, like in the previous chapter. Then we calculate

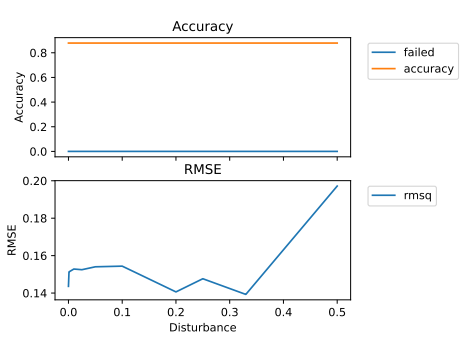
Figure 5.5: Progression of evidence resulting in changing the posterior

accuracy and root mean square errors on lower-precision networks (Figure 6.7a). We again see that while the RMSE increases when we lose precision, it does not increase by much - from 0.14 to 0.20. Rounding to ϵ and $1 - \epsilon$ instead of 0 or 1 results in robust networks that keep the same accuracy even under loss of precision. So this method works even for more complex networks that include evidence.

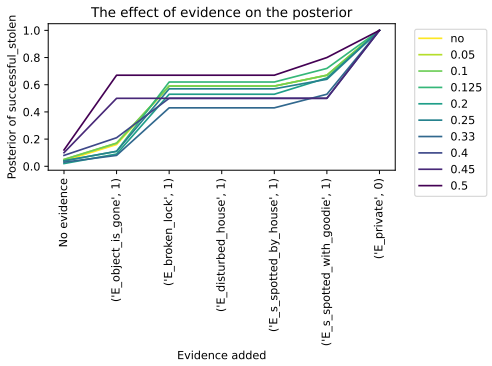
The posterior progression is also relatively the same as the original network (Figure 6.7b), only with some variation: the nuances in evidence strength are flattened-out: we see that for networks with less precision (like those rounded to 0.5 and 0.33), there are only 4 points where the value of the posterior changes, while for more precise networks, like the BN rounded to 0.05, the posterior changes value in 6 places.

2. **Could a human modeller find these probabilities?** Due to the high accuracy and similar posterior progression of the lower-precision networks, it is not essential that a modeller finds precise probabilities.

3. **Can a human modeller determine the correct independence relations?** This is more difficult - the way that evidence is connected to the hypotheses is complex and confusing. Additionally, there are sibling-nodes (nodes with the same parents), and how these nodes relate to each other temporally cannot be understood from the BN alone (does compromise house come before or after flees startled?).



(a) Network Under Disturbance.



(b) Progression of evidence resulting in changing the posterior

Figure 5.6: Loss of precision in networks.

5.1.4 Discussion

We have shown that we can model spatial simulations with agents and evidence to a reasonable accuracy (88%) in a Bayesian Network. The network generally fulfils the criteria we set out for it: The generated network is ordered temporally, complete, includes evidence, has a high accuracy, and performs well under lower precision.

Only the connections of the evidence to the network is confusing and complex. The cpts in the network can become less precise without losing accuracy, which is promising for elicitation. However, lower precision in the network reduces expression and nuance of evidence strength.

Network	Accuracy	Root Mean Square	Inconsistency error
Stolen Laptop	0.499	0.50	0

Table 5.2: Accuracy and Root Mean Square Error for the networks calculated on the impossible states.

5.2 Experiment 2: Investigating Private Knowledge

This experiment tests the effect of private knowledge on the resulting network and its accuracy and ability to get to a posterior.

5.2.1 Introduction

We can think about the process of a criminal investigation and a trial as a systemic way of gathering, sharing and finally agreeing on knowledge. Initially, only the criminal will know the entire story, victims know parts of it, and the police and judges know even less. An ideal trial results in common knowledge of the entire situation.

Of course, it is not in the interest of everyone to freely share their private knowledge all the time. Police might not want to talk about the limitations of their evidence (or their ways of procuring it), witnesses and victims might not want to talk due to incrimination, shame, or relationships to the people involved. Suspects might not want to freely share their murder plans (obviously). There is an abundance of private knowledge.

The Bayesian Network generated up till now, have not taken this into consideration. We have the full set of nodes - even nodes that a realistic investigation cannot be sure about, like the internal considerations of the criminal agent (who will flee when it sees the other agent). If we were real investigators, we cannot actually know this behaviour and model it in our network before the agent admits to the behaviour in court.

This would imply that we can only create Bayesian Networks after the fact: only when the court case is over, and all the facts and their probabilities have been established, can we start building them. Otherwise, we risk missing out on essential psychological or internal information. This would not be ideal. This leads us to ask the question: do our networks still work if we lose information about the internal intentions of agents?

5.2.2 Method

We identify the nodes 'flees_startled' and 'E_private' as events that can only be established after the fact. The reporters for these nodes are disabled and not included in the data that is passed to the K2 algorithm. This means that the network-building algorithm does not even know of the existence of these nodes, and will generate an entirely new network based on all the other nodes, except these two. The behaviour of the agents and the simulation remains the same.

5.2.3 Results

The resulting Bayesian Network is presented in Figure 5.7a. The complexity of this network is lower than the initial network because the two nodes are missing. However, even though there is information missing from the network, the accuracy and RMSE of the network, even under lower precision, are comparable to the initial network: 87.5% and 0.13 (Figure 5.7b).

The important difference between this network and the full network is the response of the posterior (Figure 5.7c). In the full network, once all the evidence was entered, we

found a posterior probability for 'stolen successfully' to be 1, for all precisions. However, in this situation, since essential evidence is missing, the posterior probability for 'stolen successfully' is reduced to 80%, and cannot be increased any further.

5.2.4 Discussion

Losing essential evidence from the network does not reduce its accuracy or increase RMSE, even under lower precision. Even though the accuracy of the network remains high when we miss private knowledge, the lack of private knowledge has important implications. We see that the probability of the posterior when all evidence is added is 0.8. It dips below the 0.95% posterior probability that is seen by Fenton et al. (2019) as the required posterior probability for determining guilt. The initial network did not have this problem. This implies that there is a fundamental uncertainty in the BN (which is true, because we are missing nodes), and we cannot be fully sure that the object was stolen. We are only fully sure when the posterior of 'stolen_successfully' goes to 1.

We are also too charitable with our interpretation of what private knowledge is. We only took 'flees startled' and the evidence for it, but in real life there are many things that are private knowledge. Nodes like 'target_object' can only be truly known when we look into the suspect's head, and while we might consider that a suspect must always have a motive, the operationalization of a 'motive' node in a real BN would be very difficult, because it is not clear how one would measure that a suspect has a motive.

5.3 Discussion

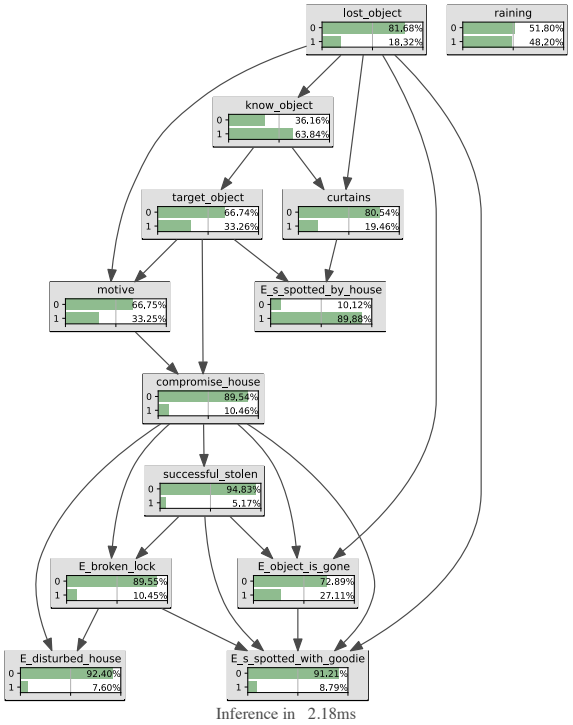
In this chapter, we have shown that we can build and experiment with Bayesian Networks that are generated from spatial multi-agent simulations with evidence. Even at lower precision, the networks perform well, although how evidence should be connected to the hypotheses remains an open question.

In these networks, we see that the evidence updates the posterior in the right direction - more evidence for stealing leads towards a higher value of 'successful-stolen'. We also see the effects of different evidence strengths - knowing that the lock is broken is more important than seeing that the house is disturbed.

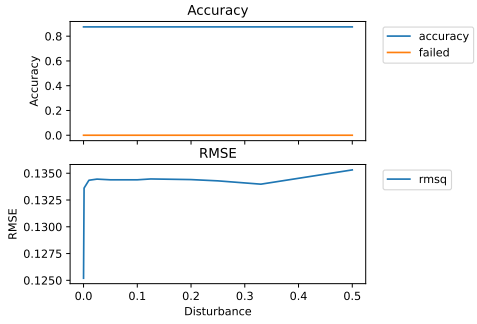
The effect of private knowledge on the generation and accuracy of the network is also more clear: it does not affect accuracy, because we can round. However, we see that we cannot get the posterior for 'stealing' as high as with the private evidence.

5.3.1 Possible Legal Interpretations.

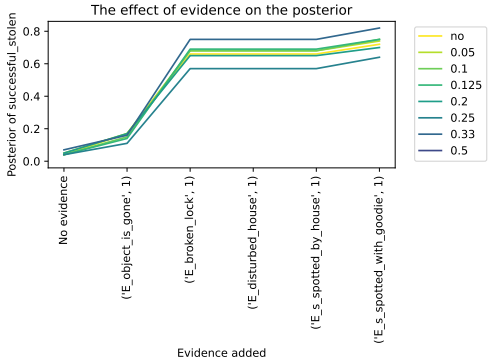
We have shown that this type of Bayesian Network does not have to be precise. This leaves room for disagreements about probabilities. We can imagine a trial where two experts have



(a) network structure for missing private information



(b) Accuracy (100% to 0%) and Root Mean Square error (1 - 0) for missing private information



(c) posterior private information

Figure 5.7: The network, the accuracy and root mean square for rounding the cpts to intervals, and the effect of evidence on the posterior for stolen laptop simulation.

to make a subjective probability estimation. Due to the finding that lower precision does not impede accuracy very much, we now have more room for disagreement. If expert one thinks that some event is 20% likely, and expert two thinks that the event is 30% likely, they might still be able to agree on 25%, and be happy with the resulting cpt for the event. Given this network, this would not reduce the accuracy of the network in predicting the outcome (although it might reduce the RMSE).

However, we run into trouble when we want to be exact about a threshold for guilt. As we have seen in comparing experiment 1 to experiment 2, we see that we cannot increase the posterior for 'stealing' in experiment 2 higher than 0.8, due to the lack of essential but private knowledge. In experiment 1, we can get this posterior to 1 with a lack of precision. We might have grounds to accept guilt in experiment 1, given the set of evidence that we showed, but do we have ground in experiment 2?

If we take for guilt a probability of 0.99, we do not, but in experiment 2 we can never reach a probability of guilt that is that high, because we miss essential information. It might be the case that if we build a network in which the final output probability can never be higher than 0.8, it could be a sign that we are missing essential information (like the lack of private knowledge), and our BN is not complete. This idea warrants further investigation.

The final and essential problem here is also something discussed in the previous chapter: the operationalisation and reference class of the events. In our simulation, we are taking this for granted, because our reporters work. We have the operationalisation, we know the reference class, and due to this we can automatically generate Bayesian Networks. However, in the real world we would find it difficult to properly operationalise many of these variables. How do we know if an agent is spotted 'by a house'? Do we mean 'we can see the agent on camera'?, but what if the camera has a large range and can see the other side of the street? How do we define 'nearness'? How do we know that our methods for determining the truth or falsity of our events are valid? How do we decide which events we are selecting in the first place? These are all open questions that are not answered by this project.

Chapter 6

Robberies

In the previous chapters, we moved from a non-spatial simulation, to a very simple spatial simulation. In the simple spatial simulations, agents move through an environment, but that environment is very trivial - there are two houses, that is about it. A simulation with non-trivial environment might lead to more uncertainty, and require more interesting agent-behaviours that might prove challenging for the Bayesian Network. Additionally, by implementing non-trivial environments, we move closer to the state-of-the-art agent models for modelling crime cases.

Research questions for this chapter:

- Can we create a simulation that somewhat reflects a real world location?
- Can we create an automatic BN from this simulation that fulfils the criteria we set out for it?

6.1 Introduction

We take the theme of the robbery from last chapter, and make it a street robbery. Two agents are walking around at the Grote Markt (the central town square in Groningen). One of the agents is always old, and always has a valuable object with them. This is the victim agent. The other agent is the thief, who is young, and always wants to steal the valuable object. If the thief sees the potential victim, they decide whether the potential victim is vulnerable (here: old), enough to target, and if the object is worth it. If these factors are both present, the agent will have a motive to steal, will attempt to sneak up on the victim, and rob them.

6.2 Methods

We wrote a method to convert screenshots of maps into an agent-readable environment. The maps were screenshotted from http://maps.stamen.com/terrain/#18/53.21618/6.57225 and converted into greyscale. Then, they were transformed into a grid of a given size. The average greyscale value of each cell in the grid was taken and coded as either 'accessible' or 'non-accessible'. 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. There are 8 cameras placed randomly on the 'accessible' cells on the map, each with a visual radius of 15. Additionally, we used this map to calculate the sight lines of both cameras and agents. An agent or a camera cannot see another agent if there is an 'inaccessible' grid cell on the sight line between the two.

The agents have some other features than before. Every agent has an age, to determine whether they are vulnerable or not - old people are considered more vulnerable. Every agent also has an object of a certain value, the thief's object has a value of -1, and the other agent's object has a value of 1000, to make it a tempting target. An agent decides if it wants to steal something by making a very simple risk-calculation based on their risk threshold: if the object is more expensive than their risk threshold, they will attempt to steal it (contributing to 'motive'). Every agent also has a goal state, this is the location at the edge of the map. When they enter their goal state, they are essentially removed from the simulation, as they leave the relevant area. Every simulation was run for 100 timesteps, or until both agents are in their goal states. The simulation itself was ran 2500 iterations. The behaviour of the agents is shown in Figure 6.1. Agents are placed randomly on the map.

The operationalisation of the random variables is described below:

seen_1_0 if the victim agent is in the line of sight of the thief agent.

know_valuable_1_0 if the value is higher than the risk threshold.

know_vulnerable_1_0 if the victim's age is older than the thief's threshold age.

motive_1_0 if the object is more valuable than the risk threshold, the victim's age is older than the thief's threshold age, and the thief is not already stealing from someone else.

sneak_1_0 if the thief has targeted the victim (has a motive) , but is not yet in the same position.

stealing_1_0 if the thief has targeted the victim, the object's value is greater than the risk threshold, and the thief's position is the same as the victim's position.

object_dropped_accidentally_0 at every epoch, there is a 1/500 probability that the victim agent will drop the object by accident.

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.

E_camera_1 the thief is seen on any one of the cameras.

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

E_camera_sees_object_1_0 if any camera sees the thief, and the thief has stolen the object, there is a 9% probability at every epoch that the thief is holding the stolen object in view of the camera.

6.3 Results

The network is shown in Figure 6.3.

6.3.1 Structural Criteria

1. **Hypotheses are ordered temporally.**

   Due to the ordering as presented to the K2 algorithm, we see that most of the hypotheses nodes from the main scenario are ordered temporally. The node 'object_dropped_accidentally_0' is the one exception. This node represents the alternative scenario where the thief does not steal, but the victim agent drops their object by accident. When the two scenarios are merged when the ordering is presented to the K2 algorithm, the second scenario is added after the first scenario, but before the evidence, because the second scenario occurs less frequently. It is correct that the node is connected to the 'motive_1_0' and 'stealing_1_0' nodes, because the thief cannot have a motive or steal an object of the victim if the victim does not possess the object anymore. Ideally, we would want this node to have no parents. That this does not happen is a flaw in the naive calculation of the temporal ordering - we should not just look at ordering, but at the time-step of the event instead to produce correct temporal ordering of merged scenarios.

2. **Evidence connects to hypotheses.**

   Every piece of evidence has at least one hypothesis node as a parent, and no piece of evidence is the parent of a hypothesis node. The network satisfies this constraint, although not every hypothesis node has evidence applied to it.

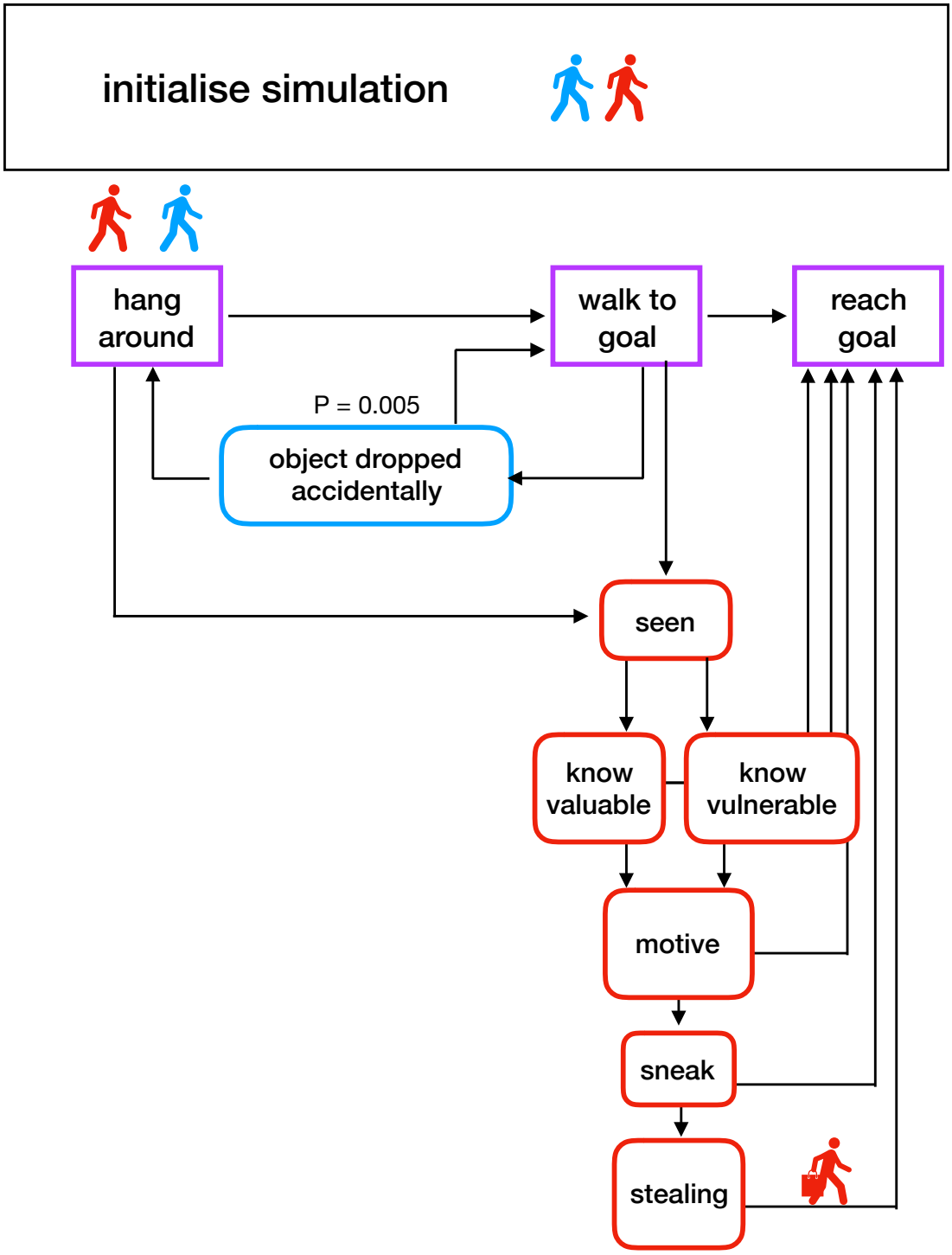
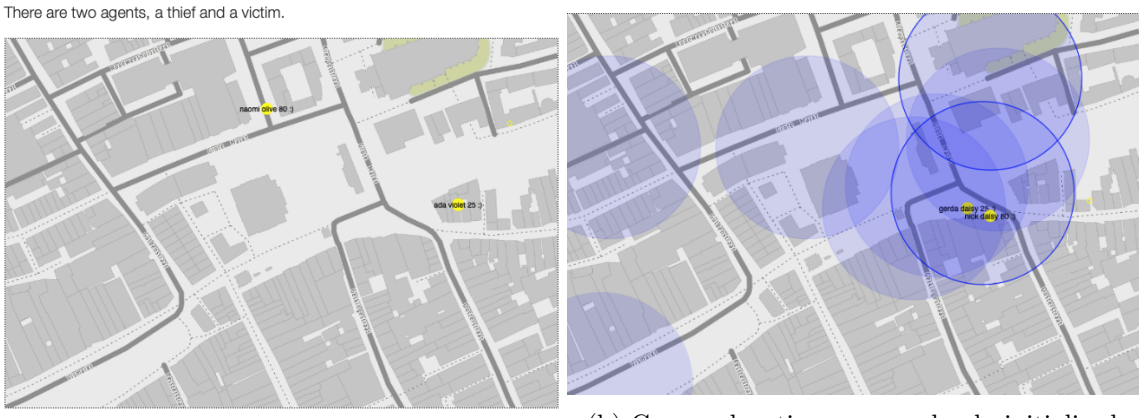


Figure 6.1: The behaviour of the agents. The nodes with rounded edges correspond to the nodes in the Bayesian Network.



(a) map of environment - 2 agents

(b) Camera locations are randomly initialized

Figure 6.2: The Grote Markt environment

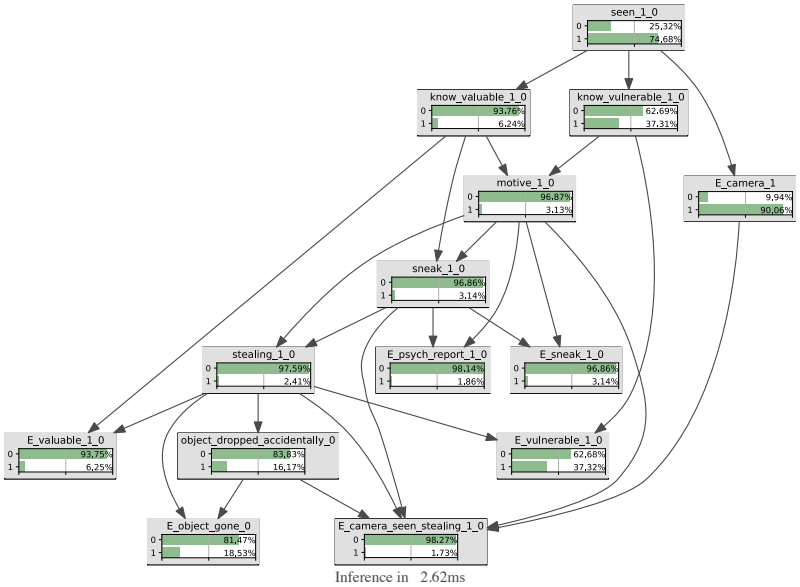


Figure 6.3: Bayesian Network

3. **Relevance: All relevant events are in the BN, all irrelevant events are outside of the BN.**

   No irrelevant events are in the network. There are some connections between nodes that seem irrelevant or overspecified, though. It is strange that 'know_valuable_1_0' is connected to 'stealing_1_0'. The thief steals iff they have a motive, and knowing that the object is valuable is only important in so far as it is relevant for the thief having a motive - there should only be a connection from motive, and not from knowing valuable. In the cpt for 'stealing_1_0' (Figure 6.4) we see that we have to specify numbers for impossible states, which we would hope to avoid as they add extra complexity.

		stealing_1_0	
motive_1_0	know_valuable_1_0	0	1
0	0	0.9999	0.0001
	1	1.0000	0.0000
1	0	1.0000	0.0000
	1	0.1687	0.8313

Figure 6.4: Conditional probability table for 'stealing_1_0'.

4. **Independent events are not connected to each other.** The parent node is not conditional on other nodes, which is correct. For the rest of the nodes, the dependency relations are complex.

6.3.2 Performance Criteria

1. **Accuracy and RMSE.**

   The accuracy of the network is 92.3%, which is better than in the previous networks. The RMSE is 0.078.

2. **Correspondence.**

   The frequency of events in the network (without evidence set) correspond to the frequency of events in the simulation within ±0.001, see Table 6.1.

3. **Sensitivity Values of Output Node.**

Conclusion	Frequency P(event)	BN P(event)
seen_1_0	0.5415	0.5415
know_valuable_1_0	0.2285	0.2286
know_vulnerable_1_0	0.5415	0.5414
motive_1_0	0.2285	0.2285
sneak_1_0	0.2285	0.2284
stealing_1_0	0.19	0.1900
object_dropped_accidentally_0	0.1515	0.1517
E_psych_report_1_0	0.1815	0.1814
E_camera_1	0.999	0.9987
E_object_gone_0	0.3415	0.3414
E_camera_sees_object_1_0	0.1890	0.1895

Table 6.1: Do we match with the frequencies?

In Figure 6.5, we see that in general, all parameters are more sensitive to changes than in the previous network. However, this network remains robust, as the largest sensitivity value that is not the same node, is only 0.45.

4. **Evidence updates the posterior in the correct direction.**

   We tell the following story with the evidence (Figure 6.6): We start with no evidence, and a posterior of 0.2 for stealing_1_0. This would not be legally correct, before we have any evidence we might have a posterior probability of something being stolen from agent 0, but we cannot know that it is agent 1 who has done the stealing. In the simulation, where only agent 1 could have stolen the object, it is correct though. Then, we add the evidence that we saw agent 1 on the camera, which is not very strong evidence, because there are a lot of cameras throughout the city. They place the agent at the scene, but are not evidence for stealing. Then, we add the evidence that the object is gone, and the posterior for stealing goes up, but does not saturate to 1, which is also correct - agent 0 could have dropped the object accidentally. Then, we have some evidence of a psychological report, that states that agent 0 fits the victim profile of agent 1, which would be some evidence, but should not be so strong, objectively, as we still have not actually seen any criminal activity from agent 1. Then, finally we see the criminal activity, and see that the camera has seen agent 1 steal the object from agent 0. The posterior for stealing goes to 1.

   We note that if we take the same story, but instead of setting 'E_camera_sees_object_1_0' to 1, we set it to 0, we find P('stealing_1_0) = 0.15, showing both that seeing the thief steal on camera is very strong evidence, and not seeing the thief steal on camera does not mean that they have not stolen (since P('stealing_1_0) $\neq$ 0). However, not seeing direct evidence of theft means that the final probability of guilt is too low to

Node	Max
seen_1_0	0.35
know_valuable_1_0	0.45
know_vulnerable_1_0	0.19
motive_1_0	0.19
sneak_1_0	0
stealing_1_0	0.77
object_dropped_acci...	0
E_psych_report_1_0	0
E_camera_1	0
E_camera_sees_objec...	0
E_object_gone_0	0

Figure 6.5: The maximum sensitivity values for every node as it relates to 'stealing_1_0' as presented in Hugin.

convict.

6.3.3 Human Criteria

1. **How robust is the network against a loss of precision?**

   We see again that the networks hold up well against loss of precision (Figure 6.7), which is promising for human builders, as long as they do not assign 1 and 0 but instead $1 - \epsilon$ and ϵ.

2. **Could a person find these probabilities?**

   Since the accuracy of the network does not suffer much under lack of precision, we have more room for error. However, it remains unclear how the probabilities should be determined. For instance, the node 'object_dropped_accidentally_0', has a posterior probability of 0.15 when no evidence is added (and two parents). This posterior probability emerges from the in-fact parameter that at any time step before reaching the goal state, the victim has a 1/500 chance of dropping the object. The time before the victim reaches the goal state depends on the victim's initial location and their goal state, both of which are random at every new iteration of the simulation. This means that it would be very difficult to trace this frequency of 0.15 back to factors that can meaningfully be elicited.

   On the other hand, nodes like 'motive' are almost constructed logically (Figure 6.8), which implies that we might not need to elicit probabilistic information about all

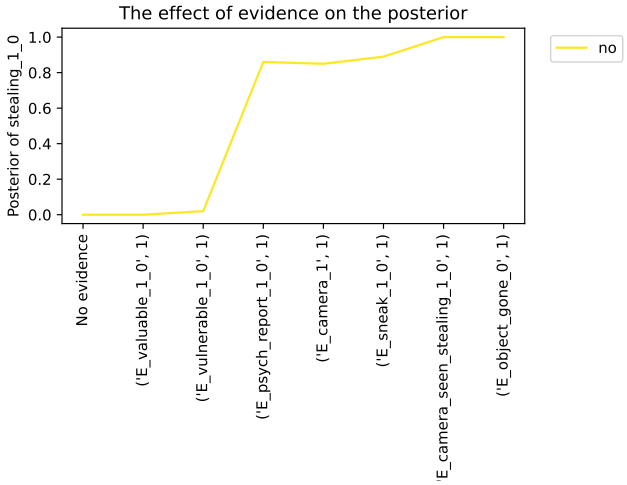
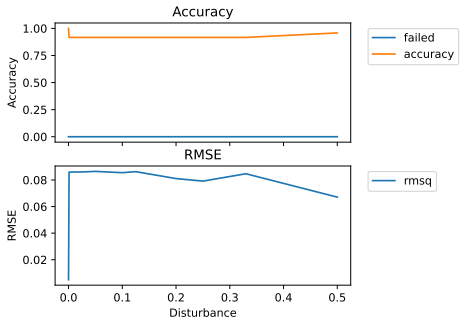
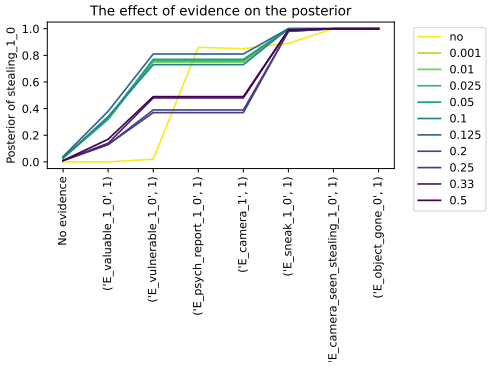


Figure 6.6: Progression of evidence resulting in changing the posterior



(a) Network Under Disturbance.



(b) Progression of evidence resulting in changing the posterior

Figure 6.7: Loss of precision in networks.

hypotheses nodes - if we find appropriate priors for nodes without parents, and their children can be constructed out of logical combinations of their parents, we might only need to elicit probabilistic information for the parents. However, most of the hypothesis nodes are not such logical combinations, and hence need effortful and almost impossible-to-find information about frequencies.

			motive_1_0	
know_valuable_1_0	seen_1_0	know_vulnerable_1_0	0	1
0	0	0	0.9999	0.0001
		1	1.0000	0.0000
	1	0	1.0000	0.0000
		1	0.9999	0.0001
1	0	0	1.0000	0.0000
		1	1.0000	0.0000
	1	0	1.0000	0.0000
		1	0.0001	0.9999

Figure 6.8: Conditional probability table for 'motive_1_0'. A logical approach.

3. **Can a person determine the correct independence relations?**

   The network has more dependence relations than would be ideal for a person to create. The ordering of the nodes would be plausible, but there are too many independence relations: 'know_valuable_1_0' has 5 children! That is not necessary, it should only have 'know_vulnerable_1_0' and 'motive_1_0' as children.

6.4 Discussion

We have shown in this chapter that we can generate satisfactory Bayesian Networks from simple agent-simulations with non-trivial environments. These networks can be used to reason towards a conclusion about a posterior probability, given a set of evidence. The network is accurate and is robust under imprecision. However, there are too many arcs in the network, and it is unclear how we should elicit the correct probabilities.

6.4.1 Limitations of the simulation.

This simulation should be seen as a preliminary for further research, as it is lacking in several aspects: the environment is still too simple, the agent-behaviour is too static, and there are only two agents in the simulation. These are the same weaknesses as identified

in (Haojie Zhu, 2021). Expanding the simulation to include or improve these aspects is useful for future research.

The non-trivial environment is still not reflecting the non-trivialness of real environments. The environment of a city offers different affordances to different people, but here the map is shared by all agents. Additionally, shops and buildings can actually be entered.

There is a lack of dynamic behaviour in the agents. They can move around and decide to steal from someone, but they do so based on simple factors. This does not meaningfully reflect the real dynamics of street crime - for example, the vulnerable agent walks around with their valuable object in plain sight, why would they do that if they know that there are thieves on the loose?

One of the purposes of modelling a 'real' location is to model the people within that real location. There are many people around on the Grote Markt in real life, modelling just two of them (and not modelling interactions with the crowd) is sufficient to show that automated Bayesian Networks might work in this situation. However, situations where Bayesian Networks might rely on statistical data from real samples (that might be taken from real locations, like the Island Prior), are not modelled in this simulation.

6.4.2 The problem of identity

We have avoided the problem of identity in this network, because we only have two agents, and one of the agents is always the thief, the other the victim. As soon as we create a more plausible simulation of crime, where everyone could, in principle, rob everyone, given the right incentives, we would need a different kind of reporter. Even if we created the same simulation, but now the victim agent might also be able to rob the thief-agent, we would need twice the amount of nodes (we need 'stealing_0_1', and the rest as well). Three agents that can each rob each other, and we have 6 times as many nodes, resulting in for n agents, $n(n-1)$ number of nodes if we use the same structure. This is very implausible due to complexity constraints. However, generalising the nodes (so using 'stealing', instead of 'stealing_0_1') means that we have to represent the identity of the suspect in some other way. It is unclear how that is to be done.

6.4.3 Implicit conditioning on environment

We used one non-trivial environment to create this network. However, the underlying geometry of the simulation affects the probabilities that can be found in the simulation. We cannot just assume one prior probability for 'seen_1_0', since the probability whether some agent sees the other, depends on the structure of the environment. We can show this by using the exact same agent behaviour, but placing the agents on a different map.

Instead of the Grote Markt, we selected 5 different parts of Groningen (Figure 6.9), con-

verted them into maps according to the method, and then let the agents loose in them to rob each other.

(a) Grote Markt.

(b) Selwerd.

(c) South-center

(d) Kattediep

(e) Street with the academy building

(f) Not Groningen, just a wall

Figure 6.9: Maps.

We look at the difference in the cpt tables for 'seen_1_0', which represents the event that agent 1 (the thief), sees agent 0 (Table 6.2). If we did not need to condition on underlying geometry of the simulation, we would expect that this probability of 'seen_1_0' would be the same, regardless of map. However, we find that the probability of the thief seeing the potential victim, depends on the underlying map.

This means that we actually cannot speak of just one 'global' probability for 'seen_1_0'. The probability of the thief seeing the victim depends on a variable that is not included in the Bayesian Network: the map. Instead of a 'global' probability for 'seen_1_0', that applies to all maps, we can only speak of the probability of the agent seeing the victim as conditioned on a specific map.

Implications of this is that we need to condition explicitly on maps for our networks to work, because it does meaningfully change the probabilities that we find. Additionally, there's no way to predict how the map that we're using affects the probability of 'seen_1_0', this probability emerges from the interaction of the agents with the map. This has implications for the real world, because it means that we can't depend on some generic "probability of getting robbed", we need to condition on spatial conditions, and background world assumptions.

map	accuracy	cpt of 'seen_1_0' True
selwerd	1	0.471
academy	0.92	0.951
groteMarkt	1	0.534
kattediep	1	0.534
wall	1	0.512
zuidCentrum	1	0.509

Table 6.2: Difference in cpt depending only on difference in underlying map, no further difference in agent behaviour!

6.4.4 Possible Legal Interpretations.

As we found in the network in the previous chapter, the problem of operationalisation continues to haunt us. The simulation has very clear operationalisation, as represented by the always-correct reporters. However, it is unclear what shape these reporters should take outside of our simulation. Lawyers and judges would have to ask themselves the following questions, to be applied to every node:

- **By what method can we find out whether this event happened or not?**

  This question can be answered relatively easily for evidence nodes. For instance, we would know that 'E_camera_1' would be true in real life, if we saw agent 1 on a relevant camera. For our psychological report, we would know that this event did not happen if the psychiatrist decided that the victim did not fit the suspect's profile.

  However, this becomes very complex for hypothesis nodes. How would you find out of 'stealing_1_0' is true in real life? In our simulation, we just set the event to 1 if it happened. In real life, how would you operationalise this? It is unclear. Should we

decide that 'stealing_1_0' is true if the victim says that their object was stolen? That's perhaps a part of an operationalisation, but it is not a completely valid one, since we know that people can lie about their stuff being stolen. If we say, the victim should say that their object was stolen, and we should find the object on the thief, we are still in trouble if the thief decides to sell the object before they can be apprehended. What would be the method to find out that the state of the real world is such that you could say that 'stealing_1_0' is true, or false?

In the real world, we do not need an operationalisation for these kinds of facts, because we all 'sort of know what we mean'. However, in Bayesian Networks we need an operationalisation, because Bayesian Networks and random variables are mathematical objects, and need to map events to truth values. This cannot be done without operationalisation.

- **If we have a method, how do we decide which events we want to include in our frequency calculation?**

  This is the problem of the reference class. To take the example of camera, if we include all cameras in the inner city, the probability that the thief would be seen is great, and hence would not be very relevant to proving the thief's guilt. However, if we decide that we will only use cameras that were on the victim's trajectory, we have limited the amount of cameras, and the probability of the thief showing up on one of these 'coincidentally' is lower, hence it would be a stronger piece of evidence if we found that it was true (and more exculpatory if we found that it was false).

  Even though the reference class is a fundamental problem, as long as every legal participant is aware of it, and can argue about why some reference class is better than another class, or decides on the appropriate classes together, this should not be a limiting factor for the Bayesian Network in itself. However, it requires a great deal of work that would not be done if Bayesian Networks were not used, and it is unsure if it would actually result in better outcomes.

These essential questions should come before talking about Bayesian Network accuracy, precision or structure, especially if these networks are to be used for practical applications.

Chapter 7

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.

7.1 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.

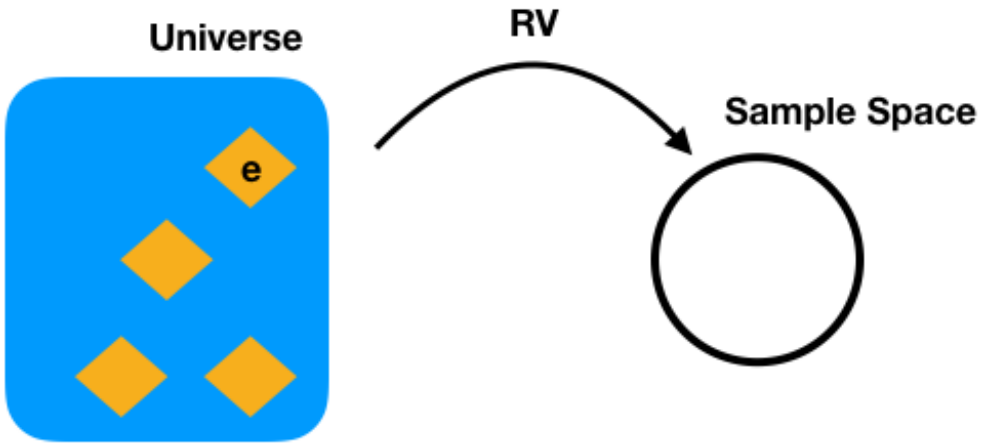
Chapter 8

Appendix

8.1 Background on probabilities - the meaning of random variables

Bayesian Networks consists of nodes, and arcs between those nodes. The nodes in a Bayesian Network are random variables. But what are random variables? The following explanation is adapted from (Jaeger, 2019).

A random variable (RV) is not just a natural language statement, but it's a mathematical object. It is a function or mapping of the form $X : \Omega \rightarrow S$, where Ω is the universe, and S is the sample space. The RV maps some elementary event in the universe, to a specific output in sample space (Figure).



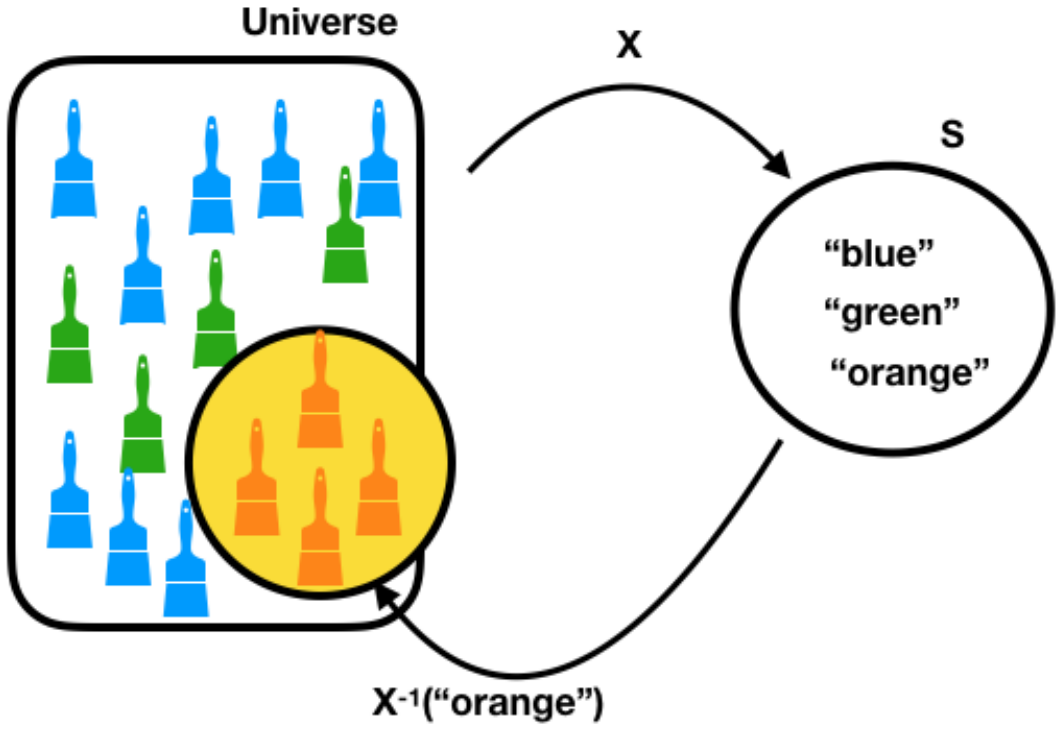
This is not really an explanation, because 1) what is a universe? 4) what is a sample space? 3) mapping how? 2) what is an elementary event? and so on.

The universe can be super abstract. It is the place where the events happen that we're interested in. It can be our actual universe, a simulation, or a subset or our actual universe or simulation. Elementary events are the events that happen within our universe - they are the elements of the set that is the universe. A random variable is a procedure, with a method, that can map an event to a certain value. All the possible values that an event can take, are collected in the sample space S. [1]

To illustrate this, we have a simple example (Example 1).

Example 1 *Let's say that we are observing a painter who wants to put down the first layer of paint on their canvas. The painter has to pick the color of the background layer.* Ω *is the universe, and contains all events where this painter is picking a color for a background for an oil painting.* $\mathcal{S}$ *is the sample space, and is the set of all colors that the painter can pick. The RV* X *maps events to values in sample space. For this initial simple example,* $S = \{green, blue, orange\}$*. Then* $X^{-1}(orange)$ *refers to the set of all events where the painter picked "orange" as the background color.*

[1] This is how far I'll go with this explanation, if you want to know about σ-fields I don't have time to understand and explain all of that. And I don't think its necessary for the problems with bayesian networks.



In our example, we say that the RV X maps events to values in sample space (and inversely, X^{-1} maps a value in sample space to a set of events). However, how does this mapping work? The mapping itself is not a mathematical object (eg, we don't do $X(\omega) = 2 * \omega^2 =$ "*orange*"). Instead, we observe a certain event w in Ω, and measure in some way, to find out what value w takes in S. For things like dice-rolls, it is clear how we observe the event (we just look at it), just like in our painting example. However, there are several ways that we can observe and map this situation, operationalise it, which all respond to different RV's:

1. **RGB Color-picker**: we take a picture of the canvas, and analyse it in the computer. We have selected certain ranges of RGB values that correspond to orange, blue and green, respectively. The average color of the canvas within one of the RGB-color bins is how we know what color it is.

2. **Taste**: we have a super-taster who specialises in paint pigments [2]. When she blind-tastes a bit of the paint, she can tell us what color it is, because the pigment that causes the color orange tastes different from the green, which tastes different from blue.

[2]Not the healthiest of occupations.

3. **Subjective taste**: I have my own opinions on whether some paint color is orange, blue or green. I see the canvas, and I decide [3].
4. Think of something else ridiculous.

Of these operationalisations of the RV you can think many things. Some might be more valid than others. We can only know if we trust the final probability assignment (I will get there in a second) if we can agree with the method of mapping - which is the random variable. The operationalisation of the random variable is a big deal, and to know if it is valid/accurate, we need to know exactly how it was mapped, because then we can argue about it if we don't like it.

Now, let's assume we have some sort of way of determining the color of the canvas (really does not matter which one, as long as we pick one). Then, we can talk about probability! Probability maps an event to a real number $[0, 1]$. The 'event' here is not the same as an elementary event. Instead, 'event' means that a RV takes some value in the sample space - so an event is the set of elementary events in the universe, for which the random variable on that event takes a specific value - $\{\omega \in \Omega | X(\omega) \in orange\}$ is the set of cases where the color on the canvas is orange.

Then, we can assign a probability value to this event, simply $P(X \in A) = p$. There's a whole debate on what p should actually mean - is it a degree of belief, or a frequency, or something else? The frequentist view is that we repeat measurements - eg, we apply the RV a lot (every time the painter is starting a new painting), and count how often the canvas is orange, and then divide that by the total amount of new canvases started, and this should be the value of p. In the subjectivist view, p can be any value as long as that value reflects our belief on how many times the painter paints orange vs blue vs green backgrounds. Then, we have our probability!

A different problem is for our universe. Are we only counting oil paintings, or are we also counting acrylics. Or did we say 'painting' when we meant 'artwork' and should we also take the painter's sketches and pastel drawings into account? This is the second problem, and it is known as the problem of the reference class. Different reference classes will result in different frequencies (also in different degrees of belief). That's why it is not just important to specify operationalisations for every random variable, but also specify exactly what part of the universe you're investigating (eg, what is and isn't in Ω).

[3] I might be colorblind

Bibliography

Allen, R. and Pardo, M. (2007). The problematic value of mathematical models of evidence. *The Journal of Legal Studies*, 36(1):107–140. 7

Bex, F. (2010). *Arguments, Stories and Criminal Evidence*. Springer Science and Business Media LLC. 5

Bosse, T. and Gerritsen, C. (2008). Agent-based simulation of the spatial dynamics of crime: On the interplay between criminal hot spots and reputation. In *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems*. 7

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. 12

Colyvan, M. (2013). Idealisations in normative models. *Synthese*, 190(8):1337–1350. 8

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. 7

Cooper, G. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347. 12

Dahlman, C. (2020). De-biasing legal fact-finders with bayesian thinking. *Topics in Cognitive Science*, 12(4):1115–1131. 6

Dawid, A. P. (2008). Beware of the dag! In *JMLR: Workshop and Conference Proceedings 6: 59–86*. 2

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. 6

Druzdzel, M. J. and van der Gaag, L. C. (2000). Building probabilistic networks: awhere do the numbers come from? *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*. 7

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. 6

Fenton, N., Neil, M., and Lagnado, D. (2011). Modelling mutually exclusive causes in bayesian networks. *Submitted to IEEE Transactions on Knowledge and Data Engineering*. 32

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. 5, 6, 13, 41

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. 5, 6, 48

Gerritsen, C. (2015). Agent-based modelling as a research tool for criminological research. *Crime Science*, 4(1). 7

Haojie Zhu, F. W. (2021). An agent-based model for simulating urban crime with improved daily routines. *Computers, Environment and Urban Systems*. 7, 61

J Kadane, D. S. (1996). *A Probabilistic Analysis of the Sacco and Vanzetti Evidence*. John Wiley and Sons. 6

Jaeger (2019). Principles of statistical modelling. 69

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. 7

Oniśko, A. and Druzdzel, M. J. (2013). Impact of precision of bayesian network parameters on accuracy of medical diagnostic systems. *Artificial Intelligence in Medicine*. 15

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann. 2

Pennington, N. and Hastie, R. (1993). 5

Renooij, S. (2001). Probability elicitation for belief networks: issues to consider. *The Knowledge Engineering Review*, 16(3):255–269. 7

Renooij, S. and Witteman, C. (1999). Talking probabilities: communicating probabilistic information with words and numbers. *International Journal of Approximate Reasoning*, 22(3):169–194. 27

Ronald Meester, K. S. (2021). *Probability and Forensic Evidence*. Cambridge University Press. 6

Timmer, S. (2016). *Designing and Understanding Forensic Bayesian Networks using Argumentation*. PhD thesis, University of Utrecht. 5

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. 15

van Leeuwen, L. (2019). A comparison of two hybrid methods for analysing evidential reasoning. In *JURIX*. 6

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. 5

Vlek, C. (2016). *When stories and numbers meet in court*. PhD thesis, University of Groningen. 5

Vlek, C. S., Prakken, H., Renooij, S., and Verheij, B. (2014). Building bayesian networks for legal evidence with narratives: a case study evaluation. *Artificial intelligence and law*, 22(4):375–421. 31

Vlek, C. S., Prakken, H., Renooij, S., and Verheij, B. (2015). Representing the quality of crime scenarios in a bayesian network. In *JURIX*. JURIX. 3, 5, 17, 18, 19, 30, 31

Wagenaar, W. A., Van Koppen, P. J., and Crombag, H. F. (1993). *Anchored narratives: The psychology of criminal evidence*. St Martin's Press. 5