Gated Bayesian Networks for Algorithmic Trading

Marcus Bendtsen
marcus.bendtsen@liu.se

Jose M. Peña
jose.m.pena@liu.se

Department of Computer and Information Science, Linköping University, Sweden

Abstract

This paper introduces a new probabilistic graphical model called *gated Bayesian network* (GBN). This model evolved from the need to represent processes that include several distinct phases. In essence, a GBN is a model that combines several Bayesian networks (BNs) in such a manner that they may be active or inactive during queries to the model. We use objects called *gates* to combine BNs, and to activate and deactivate them when predefined logical statements are satisfied. In this paper we also present an algorithm for semi-automatic learning of GBNs. We use the algorithm to learn GBNs that output buy and sell decisions for use in *algorithmic trading* systems. We show how the learnt GBNs can substantially lower risk towards invested capital, while they at the same time generate similar or better rewards, compared to the benchmark investment strategy *buy-and-hold*. We also explore some differences and similarities between GBNs and other related formalisms.

1 Introduction

Bayesian networks (BNs) can be interpreted as models of causality at the macroscopic level, where unmodelled causes add uncertainty. Cause and effect are modelled using random variables that are placed in a directed acyclic graph (DAG). The causal model implies some probabilistic independencies among the variables, that can easily be read off the DAG. Therefore, a BN does not only represent a causal model but also an independence model. The qualitative model can be quantified by specifying certain marginal and conditional probability distributions so as to specify a joint probability distribution, which can later be used to answer queries regarding posterior probabilities, interventions, counterfactuals, etc. The independencies represented in the DAG make it possible to compute these posteriors efficiently. Furthermore, they reduce the number of parameters needed to represent the joint probability distribution, thus making it it easier to elicit the probability parameters needed from experts or from data. See [1, 2, 3] for more details.

A feature of BNs, known as the local Markov property, implies that a node is independent of all other non-descendent nodes given its parent nodes, where the relationships are defined with respect to the DAG of the BN. If we define the parents of X_i as $Parents(X_i)$, the local Markov property allows us to factorise the joint probability distribution according to Equation 1.

$$p(X_1, X_2, ..., X_n) \;=\; \prod_{i=1}^{n} p(X_i|Parents(X_i)) \tag{1}$$

Despite their popularity and advantages, there are situations where a BN is not enough. For instance, when trying to model the process of a trader buying and selling stock shares, we wanted a model that switched between identifying buying opportunities and then, once such have been found, identifying selling opportunities. The trader can be seen as being

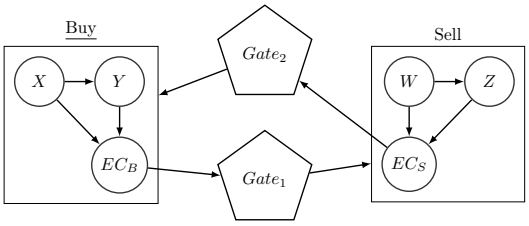

Figure 1: GBN using two phases

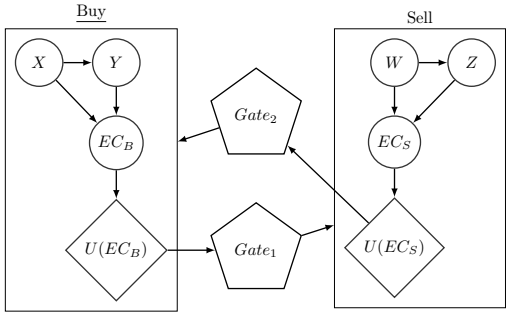

Figure 2: GBN using utility nodes

in one of two distinct phases: either looking for an opportunity to buy shares and enter the market, or an opportunity to sell shares and exit the market. These two phases can be very different and the variables included in the BNs modelling them are not necessarily the same. Dynamic BNs have traditionally been used to model temporal processes, and as their name suggests, they model the dynamics among variables between typically equally spaced time steps. However, processes that entail different models at different phases, and where the transition between phases depend on the observations made, are not easily captured by dynamic BNs, as they assume the same static network at each time step. The need to switch between different BNs was the foundation for the probabilistic graphical model presented herein, which we call gated Bayesian networks (GBNs). In Figure 1 we present a GBN that uses two different BNs (*Buy* and *Sell*). In Section 2.2 we will explain how decisions can be connected to the phase changes of a GBN, we will specifically show how buy and sell decisions are connected to the phase changes for the GBN in Figure 1. It should however be noted that we will not always connect a phase change with a decision, as there will be an example of in Section 3.2. Sometimes a phase change is needed in order to use a different BN without any explicit decision connected to it.

Intuitively, a GBN makes explicit the possible transitions between the contained models, i.e. the phases, along with the driving variables in these phases. This is not only advantageous from a representational point of view, but since constraints are encoded in the model, parameter learning will be influenced by these constraints. For instance, when a transition from the *Sell* BN in Figure 1 should occur will be dependent on when a transition from the *Buy* BN occurs, as one must happen before the other. Imagining two experts, where one gives recommendations of when to buy assets and the other when to sell assets, we would want the experts to work well together. If the first expert has a long-term view and the second expert has a short-term view, then recommendations to buy will be far apart, but as the second expert assumes that we are after short-term profits, sell recommendations come quickly after we have bought the assets. In extreme cases, this may end up in a strategy where over a year the assets are only held for a few hours. Thus, the fact that buying and selling places constraints on each other must be captured by the model, and single BNs are not able to encode these constraints.

The example of the trader is really a simplification of a more complex process known as *algorithmic trading*, which we will describe in more detail in the coming section. Our primary intention is to use GBNs as part of algorithmic trading, however for clarity, we will sometimes fall back to the more simple view of a single trader in this paper.

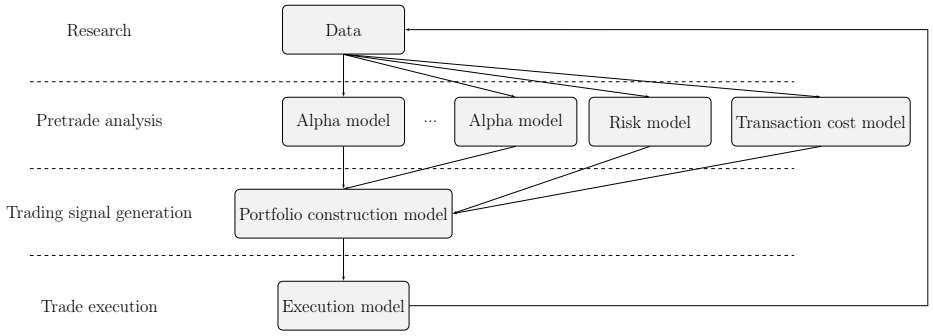

Figure 3: Components of an algorithmic trading system

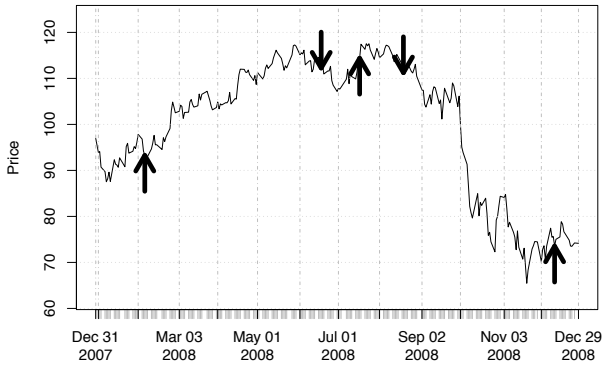

Figure 4: Buy and sell signals

1.1 Algorithmic Trading

Formally, the process we intend to model is part of a larger process commonly referred to as algorithmic trading. Algorithmic trading can be viewed as a process of actively deciding when to own assets and when to not own assets, so as to get better *risk* and *reward* on invested capital compared to holding on to the assets over a long period of time. At the other end of the spectrum is the *buy-and-hold* strategy, where one owns assets continuously over a period of time without making any decisions of selling or buying during the period.

An algorithmic trading system consists of several components, some of which may be automated by a computer, and others that may be manually executed [4, 5, 6]. A schematic overview of the components of a general algorithmic trading system is shown in Figure 3.

The type of data used at the research stage varies greatly, e.g. net profit, potential prospects, sentiment analysis, analysis of previous trades, or *technical analysis*, which will be the focus in the included application. The analysis of the data is split up into *alpha*, *risk* and *transaction cost* models. The alpha models are responsible for outputting decisions for buying and selling assets based on the data they are given. These decisions are known as buy and sell *signals*, examples of which are depicted in Figure 4 (an arrow pointing upwards is a buy signal and a downwards facing arrow is a sell signal, the signals are drawn on top of the assets historical price). If followed, these buy and sell signals give rise to certain risk and reward on the initial investment (which will be described further in Section 5.1).

The risk and transaction cost models should be seen as strategies for managing risk and transaction costs in a system that has many alpha models. The output from these three types of models (alpha, risk and transaction) are in their turn the input to the *portfolio construction model* in the trading signal generation stage. Here the output of the previous components are combined to decide which signals to actually execute in order to create a portfolio that is based on a combination of alpha models. These signals can lead to decisions to buy more of a certain asset, to sell all or a portion of assets already owned, or in some cases to *short* certain assets so that reward is achieved when the asset loses value. Portfolio construction is a widely researched topic that has been approached from both the financial field and from an information theoretic perspective. In finance, the most common basis is the theory of mean-variance portfolios, also known as Markowitz portfolio theory [7], where the tradeoff between expected risk and reward is used to allocate resources amongst a basket of assets. While from an information theoretic background the focus has been on *online* porfolio construction algorithms [8], such as the universal portfolio [9] and the exponential gradient [10], where resources are initially allocated equally, but sequentially are changed according to varying criteria that in the long run creates optimal growth.

The final stage is the actual *execution* of the trading signals, which must be done in a manner that does not affect the price of the asset that is being bought. Although all components are important, we will not be addressing all of them in this paper, instead our contribution is concerned with the use of GBNs as alpha models (informally the trader buying and selling shares can be seen as an alpha model).

The rest of this paper is disposed as follows. In Section 2, we will define the structural semantics of GBNs through a set of definitions. We will also define how GBNs can be used in a decision making context, as well as defining how the model is executed over a set of data. Section 3 gives a detailed example of the modelling and execution of a GBN, as well as an example of a GBN used in another domain than the one from the initial motivation. In Section 4 we introduce an algorithm that can be used to semi-automatically learn GBNs, followed by a real-world application of this learning algorithm in Section 5. We will offer a comparison of GBNs to other models and formalisms in Section 6, highlighting key differences and similarities. Finally, we will end this paper with our conclusions of the current work and our thoughts about future work in Section 7.

This paper unifies previous conference papers [11, 12] and extends upon them with the introduction of utility nodes in GBNs, as well as an additional experiment using GBNs that contain utility nodes. Furthermore, examples have become more detailed and comparisons to related models and formalisms have been added and extended.

2 Definitions and Model Execution

Supported by the definitions in this section, we will describe the structural semantics of GBNs, how GBNs can be used in a decision making context, as well as defining how a GBN is executed. A GBN models a sequential process, driven by an ordered set of evidence, thus it is natural to think of some index that identifies a unique position along the process. We will use t to define a unique time in a temporally ordered set of evidence. It is worth mentioning that evidence can be recorded at irregular times, thus the time interval between $t-1$ and t can be different than t and $t+1$. While reading the definitions in this section, it may be helpful to use the example GBNs offered in Figure 1 and Figure 2 as reference. In Section 3 we will give two examples of GBNs that clarify and put into context the definitions

of this section.

2.1 Structural Definitions

A GBN is a probabilistic graphical model that combines multiple BNs using objects called gates, in order to model processes that have several distinct phases. These gates allow for activation and deactivation of the different BNs in the model. Inference is carried out in the currently active BNs, thus they are participating in the current phase.

Definition 1 (GBN) *A GBN consists of a set of gates $\mathcal{G}$, a set of BNs $\mathcal{B}$ and a set of directed edges $\mathcal{E}$ that connect the gates with the BNs. Let $\mathcal{B}^A$ be the set of active BNs and $\mathcal{B}^I$ the set of inactive BNs. $\mathcal{B}^A$, $\mathcal{G}$ and $\mathcal{E}$ cannot be empty. A BN cannot belong to both $\mathcal{B}^A$ and $\mathcal{B}^I$ at the same time. Each BN consists of a set of nodes (chance and utility nodes[1]) and a set of directed edges.*

$$
\begin{aligned}
&GBN = \{\mathcal{G}, \mathcal{B}, \mathcal{E}\} \\
&\mathcal{B} = \mathcal{B}^A \cup \mathcal{B}^I, \mathcal{B}^A \cap \mathcal{B}^I = \emptyset \\
&\mathcal{B}^A, \mathcal{G}, \mathcal{E} \neq \emptyset \\
&V(\mathcal{B}_i) = \{\textit{all chance nodes in } \mathcal{B}_i\} \ , \ \mathcal{B}_i \in \mathcal{B} \\
&U(\mathcal{B}_i) = \{\textit{all utility nodes in } \mathcal{B}_i\} \ , \ \mathcal{B}_i \in \mathcal{B} \\
&E(\mathcal{B}_i) = \{\textit{all edges in } \mathcal{B}_i\} \ , \ \mathcal{B}_i \in \mathcal{B}
\end{aligned}
$$

Thus, the sets $\mathcal{B}^A$ and $\mathcal{B}^I$ from Definition 1 may contain different BNs at different times t. As mentioned earlier, at a given time t, inference is carried out in the BNs in $\mathcal{B}^A$, thus they are participating in the current phase of the process and are partially responsible for whether the process stays in the same phase or moves to another phase. When drawing a GBN, all BNs that are active prior to any evidence being supplied to the model have their names underscored (i.e. the initial set $\mathcal{B}^A$). In Figure 1 for instance, *Buy* is active prior to any evidence being supplied.

Definition 2 (Connections) *The directed edges $\mathcal{E}$ connect either a node in $V(\mathcal{B}_i)$ or $U(\mathcal{B}_i)$ with a gate in $\mathcal{G}$, or a gate in $\mathcal{G}$ with an entire BN in $\mathcal{B}$. An edge between a node and a gate is always directed away from the node towards the gate. An edge that connects a gate with an entire BN is always directed away from the gate towards the BN.*

Definition 3 (Parent/child) *When a node is connected to a gate we consider the BN to which the node belongs to be a parent of the gate. When an entire BN is connected to a gate we consider the BN to be a child of the gate.*

In Figure 1, two of the type of edges in $\mathcal{E}$ are represented, for instance the edge from chance node EC_B to $Gate_1$ implies that the *Buy* BN is a parent of $Gate_1$ while the edge from $Gate_1$ to the BN *Sell* implies that *Sell* is a child of $Gate_1$. The third and final type of edge in $\mathcal{E}$ is represented in Figure 2, the edge from utility node $U(EC_B)$ implies that *Buy* is a parent of $Gate_1$. Definition 2 and Definition 3 also allow for a temporal order semantic to be given to the edges in $\mathcal{E}$. A process moves in the direction of the edges, where the gates define points where certain criteria must be met until the process can continue. Therefore, it is the evidence available at time t, together with the BNs in $\mathcal{B}^A$ and the gates that decide if the process stays in the current phase, or moves into a new phase in $t+1$. How the criteria in the gates are defined and met is explained in the following two definitions.

[1] BNs that are extended with utility and decision nodes are usually known as influence diagrams. We do not adopt the entire framework of influence diagrams, we only use the utility node to map variables' states to real values. Therefore we use the term BN rather than influence diagram.

Definition 4 (Trigger node) *A node that is connected with a gate is called a trigger node. All nodes that are connected to a gate make up the gate's trigger nodes. It follows from Definition 2 that all gates are children of their trigger nodes.*

Definition 5 (Trigger logic) *Each trigger node of a gate $\mathcal{G}_i$ in $\mathcal{G}$, that belongs to a BN in $\mathcal{B}^A$, supplies a value to the gate each time new evidence is entered into the model. If a trigger node belongs to a BN in $\mathcal{B}^I$, then the trigger node will not supply any value. Each gate has its own trigger logic, denoted as $TL(\mathcal{G}_i)$. The trigger logic is a logical statement regarding the values that the trigger nodes supply. Specifically, the values that are supplied are:*

- *For trigger nodes that are chance nodes: the posterior probability of the random variable taking a specific value, given some evidence.*
- *For trigger nodes that are utility nodes: the utility values weighted by the joint posterior distribution of the utility node's parents, given some evidence.*

Definitions 4 and 5 complete the structural definitions by defining how the criteria for the process to move forward are formed. Exactly how this is executed will be described in Section 2.3. However, it should be clear that the BNs that at time t are in $\mathcal{B}^A$ are driving the current phase, supplying values to the gates, and when the trigger logic for one or more gates is satisfied, the temporal process moves forward to another phase. For instance, EC_B is a trigger node for $Gate_1$ in Figure 1, and assuming that EC_B has some state *positive*, $Gate_1$ could define its trigger logic as: $TL(Gate_1) : p(EC_B = positive|\mathbf{e}_t) > \tau$, where $\mathbf{e}_t$ is the evidence available at time t and τ is some threshold. It is also possible to use a utility node as a trigger node. In Figure 2 the GBN from Figure 1 has been altered to use utility nodes. These nodes map states from EC_B and EC_S to utilities, thus quantifying the value of a positive and negative climate. The trigger logic of the gates are then statements of the utility values weighted by the joint posterior distribution of the parents of the utility nodes. For instance, assuming instead that EC_B has six different states $i = 1, ..., 6$ then summing up the weighted utilities, we can require the expected utility to be higher than some threshold, $TL(Gate_1) : \sum_{i=1}^{6} p(EC_B = i|\mathbf{e})u(EC_B = i) > \tau$ (in the discrete case).

2.2 Strategy Encoding and Decisions

The structural definitions in Section 2.1 allows us to view GBNs as encoding a *strategy*. This strategy will be followed as evidence is presented to the model (exactly how will be explained in Section 2.3). In order to clarify this, let Φ be the set of every possible evidence set that can be presented to the GBN (i.e. it is the set of every possible configuration of the variables in the BNs in $\mathcal{B}$). The trigger logic of each gate then maps each set in Φ to either true or false, given the current BNs in $\mathcal{B}^A$. Specifically, let δ_i be the mappings that the trigger logic of gate i defines, we then have $\delta_i(\mathcal{B}^A, \Phi) = \{true, false\}$. We can then define the strategy that a GBN encodes as $\Delta = \{\delta_i, i = 1, ..., n\}$, where n is the number of gates in the GBN. It is then clear that a GBN only encodes a strategy for when to trigger gates.

GBNs are not strictly decision models; possible decisions, actions and potential outcomes are not made explicit in the model. However, it is possible to map the strategy that a GBN encodes to a set of decisions, e.g. for the GBN in Figure 1 we can define which decision to take by the *decision function* in Equation 2. In this example we map each evidence set $\mathbf{e}$ that we observe to a decision, given the current active BNs.

$$Decision|\mathbf{e} = \begin{cases} \text{Buy if } \delta_1(\mathcal{B}^{\mathcal{A}}, \mathbf{e}) \\ \text{Sell if } \delta_2(\mathcal{B}^{\mathcal{A}}, \mathbf{e}) \\ \text{Do nothing if none of the above apply} \end{cases} \tag{2}$$

An equivalently way of defining this decision function, one that is more manageable from an operational standpoint, is to say that given a set of triggered gates, we return a decision depending on which gates that triggered, as in Equation 3. Since GBNs allow for multiple gates to trigger at the same time, consideration of such cases must be taken when defining the decision function. For instance, the function in Equation 3 could be expanded to also state that if both $Gate_1$ and $Gate_2$ triggers, then it should be considered as a signal that the model is ambivalent regarding the market, and thus no decision should be made at all.

$$Decision|triggered\ gates = \begin{cases} \text{Buy if } Gate_1 \in triggered\ gates \\ \text{Sell if } Gate_2 \in triggered\ gates \\ \text{Do nothing if none of the above apply} \end{cases} \tag{3}$$

2.3 Model Execution

Having defined the structural definitions and explained how decisions can be read from the GBN, we continue by explaining how the model is to be executed over a set of data. Here we will offer one additional definition that is an integral part of the execution, and then define an execution algorithm that formalises how evidence is sequentially entered into the model, and how the model reacts given the evidence.

Definition 6 (Triggering, activation and deactivation) *If evidence is supplied to a GBN that leads to the trigger logic for some gate being satisfied, then the gate is said to trigger. When a gate triggers, it activates all its child BNs and deactivates all its parent BNs. If several gates trigger due to the same set of evidence then the union of all child BNs are activated and the union of all parent BNs minus the union of all child BNs are deactivated.*

$$\begin{aligned} &UCBN \ = \ Union\ of\ all\ child\ BNs\ to\ triggered\ gates \\ &UPBN \ = \ Union\ of\ all\ parent\ BNs\ to\ triggered\ gates \\ &BNs\ to\ activate \ = \ UCBN \\ &BNs\ to\ deactivate \ = \ UPBN \setminus UCBN \end{aligned}$$

Figure 5 represents a high-level outline of the execution algorithm, a detailed description of the algorithm will be given in Section 2.3.1. Given a set of sequential evidence sets $[\mathbf{e}_1, ..., \mathbf{e}_t, ..., \mathbf{e}_T]$, the algorithm starts by instantiating the variables of all active BNs with the first evidence set $\mathbf{e}_1$. As was mentioned in the comment to Definition 1, which BNs that are initially active is defined when the model is created. The trigger logic for each gate is then checked, and if it is satisfied for any of the gates, the child BNs of these gates are activated (according to Definition 6). If any BNs were activated, then the algorithm goes back and instantiates all variables of active BNs with the current evidence set, checks the trigger logic and activates BNs. Once the previous loop does not result in any new BNs being activated, all parent BNs of triggered gates that are not child BNs of triggered gates are deactivated (according to Definition 6). GBNs are allowed to contain cycles, however as deactivations only occur once all activations have been handled, the execution algorithm will always terminate, and no infinite loops will be created. If a decision function has been

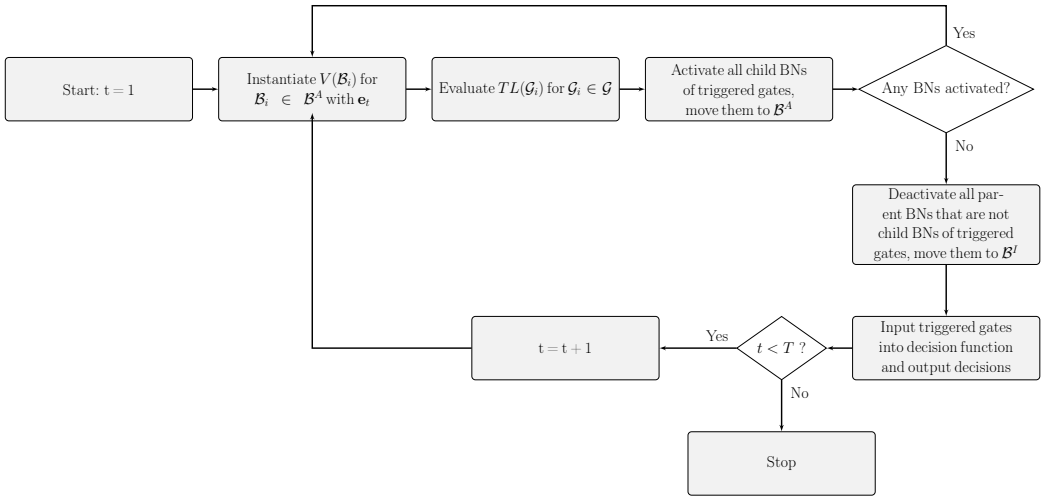

Figure 5: High-level outline of execution algorithm

defined, then the set of triggered gates are used as input to the decision function, and any returned decisions are executed.

If there exists more evidence sets, then t is incremented and the next evidence set is processed. The active/inactive state of each BN is remembered between evidence sets. Variables in inactive BNs are never instantiated with new evidence. A variable is instantiated with some evidence until a new evidence set instantiates it to a different state, thus evidence is never retracted from variables. Once no more evidence sets exist, the execution algorithm terminates.

2.3.1 Execution algorithm

In Figure 6, a detailed description of how the execution algorithm processes a sequentially ordered dataset $\mathcal{D}$ is given. On line 2, the outer loop starts that picks out the current evidence set $\mathbf{e}_t$ and passes it to the function EVIDENCE. The result of the function call is a set of gates that triggered due to $\mathbf{e}_t$. These are used as input to an externally defined decision function on line 4 (as discussed in Section 2.2), which returns the decisions to take. On line 13, inside function EVIDENCE, the inner loop of the algorithm starts. In each iteration, variable instantiations are updated for all active BNs. Variables that were previously instantiated, but for which no new evidence has been supplied, keep their instantiation. The algorithm then finds those gates that have not yet triggered and sends them to the TRIGGER function on line 30. The function will loop over the gates that have not yet triggered, evaluate their trigger logic, and if it is satisfied, adds the gate to the set of triggered gates. This set of triggered gates is then returned to the calling function and these will be added to the set ATG, which contains all the triggered gates. For each of the gates that triggered during this iteration of the loop (that started on line 13) their parent and child BNs are stored. Before the loop starts again, all child BNs that belong to triggered gates are activated. This is done in order to not enforce any ordering of the gates, so we can check the trigger logic for the gates in any order, and the same gates will trigger regardless. As long as there are gates that trigger the loop will continue. Once the loop is done, all

BNs that are parents of gates that triggered, but are not children of any triggered gates, are deactivated. The deactivation is done outside the loop for the same reasoning of unordered gates previously mentioned. Finally all triggered gates are returned.

Notice that on line 17 we are creating a set of gates that belong to the GBN but have not yet triggered. It is this set of gates that are sent to the TRIGGER function on line 18. So once a gate has triggered it cannot be triggered again. Therefore the algorithm will always terminate, if not before then at least once all gates have triggered. This prevents any form of oscillation or infinite loop of triggering gates to happen.

Variables that are in inactive BNs will not be instantiated with new evidence. Since it is difficult for the user to predict which BNs will be activated on line 24, it is important that all available evidence is given to the model each time new evidence is made available, even for variables for which the evidence might not have changed since the last set. For instance, assume that a variable A belongs to an inactive BN at time $t = 1$. At this time new evidence is observed for A, however since it does not belong to an active BN it will not be instantiated with this new evidence. Now assume that at $t = 2$, the BN that A belongs to has become active but the available evidence for A has not changed since $t = 1$. Even though the evidence has not changed for A between $t = 1$ and $t = 2$, it should be supplied to the model as it should not be required by the user to remember which BNs that have been inactive at which times in the past.

3 Execution and Modelling Examples

In order to put into context the definitions presented in Section 2, we will in this section demonstrate the application of the execution algorithm in the domain of the initial motivation for GBNs. We will also give an illustrative example of how GBNs can be used in a different domain than algorithmic trading, and show the potential of GBNs as the number of phases increase.

3.1 The Trader's Problem

The trader's problem is the scenario, in its simpler form, that initially motivated us to define GBNs, and a precursor to the real-world application that will be presented in Section 5. Assume that a trader wants to buy shares of a company when there is a belief that the share price will increase (i.e. there is a positive economical climate for this company). If the trader owns shares of the company then the trader wants to sell the shares if there is a belief that the share price will decrease (i.e. there is a negative economical climate for this company). The trader's problem is to decide when to move back and forth between the two phases of buying and selling shares, in such a way that it benefits the trader (what constitutes to being beneficial will be discussed in Section 5.1.2). The general problem solved by portfolio construction, such as the universal portfolio or Markowitz portfolio, is allocation of resources to several assets. The problem posed here is therefore slightly different, as we are only considering a single asset.

The scenario can be modelled using the GBN depicted in Figure 1. Here, X and Y are some features that predict the economical climate EC_B during the identification of buying opportunities. Similarly, W and Z predict the economical climate EC_S during the identification of selling opportunities. While variables EC_B and EC_S may be representing the same phenomenon, the posterior for these two variables will be computed at different

```
1: function EXECUTE(GBN, 𝒟)                                   ▷ 𝒟 contains all evidence sets
2:     for e_t ∈ 𝒟 where t = 1, ..., T do
3:         triggered gates ← EVIDENCE(GBN, e_t)
4:         decisions ← Decision(triggered gates)              ▷ Externally defined function
5:         execute decisions                                  ▷ Act upon the decisions generated
6:     end for
7: end function
8:
9: function EVIDENCE(GBN, e)                                  ▷ e is a set of evidence
10:    UCBN ← { }                                             ▷ children BNs of triggered gates
11:    UPBN ← { }                                             ▷ parent BNs of triggered gates
12:    ATG ← { }                                              ▷ all gates that triggered due to e
13:    repeat
14:        for all B_i ∈ B^A do
15:            Instantiate V(B_i) according to e
16:        end for
17:        NotTriggered ← G \ ATG
18:        Triggered ← TRIGGER(NotTriggered)
19:        ATG ← ATG ∪ Triggered
20:        for all G_t ∈ Triggered do
21:            UCBN ← UCBN ∪ children of G_t
22:            UPBN ← UPBN ∪ parents of G_t
23:        end for
24:        activate all B_i ∈ (UCBN)
25:    until Triggered is empty
26:    deactivate all B_i ∈ (UPBN \ UCBN)
27:    return ATG
28: end function
29:
30: function TRIGGER(NotTriggered)
31:    Triggered ← { }
32:    for all G_i ∈ NotTriggered do
33:        trigger ← EVALUATE(TL(G_i))
34:        if trigger then
35:            Triggered ← Triggered ∪ {G_i}
36:        end if
37:    end for
38:    return Triggered
39: end function
40:
41: function EVALUATE(TriggerLogic)
42:    Return evaluation of TriggerLogic. This evaluation includes posterior probability
       queries to appropriate BNs and utility calculations, as explained in Definition 5.
43: end function
```

Figure 6: Execution algorithm

times with different evidence (we use subscripts do differentiate between the variables as they are present in both BNs). Also, EC_B and EC_S represent future states, thus they would be unobservable in a real setting. The variables X, Y, W and Z come before the unobservable variables in temporal order, therefore the edges are directed away from the observed variables towards the unobserved variables. This allows us to directly model the conditional probabilities $p(EC_B|X,Y)$ and $p(EC_S|W,Z)$. However, this is only tractable if very few observed variables are considered, if the number of observed variables were to increase, then alternatives should be explored in order to reduce the number of parameters in the model, for instance by using BN classifiers [13].

$Gate_1$ is programmed with trigger logic that defines when the trader wants to buy shares, in this example we will use $TL(Gate_1) : p(EC_B = positive|\mathbf{e}) > 0.8$, where $\mathbf{e}$ is evidence. $Gate_2$ defines when the trader wants to sell shares, in this example we will use $TL(Gate_2) : p(EC_S = negative|\mathbf{e}) > 0.6$. A line under the name of the BN Buy indicates that it is active prior to any evidence being entered into the model. We will use the decision function in Equation 3

As is evident, the two decisions to buy and sell shares are dependent on different features (X, Y, W and Z). Furthermore, we can program the trigger logic in such a way that we can be more sensitive to negative climate (using a lower threshold of 0.6) and less sensitive to positive climate (using a higher threshold of 0.8). This is one way of modelling the trader's preferences.

Assume that all variables are binary and that the following evidence sets will be presented to the model:

- Set 1: X = 1, Y = 0
- Set 2: X = 1, Y = 1, W = 0
- Set 3: X = 1, Y = 0, W = 0, Z = 1
- Set 4: X = 1, Y = 0, W = 1, Z = 1

The execution algorithm will then work as follows:

- Set 1: Variables X and Y belong to an active BN Buy, and so they are instantiated according to the evidence. However, assume that this infers $p(EC_B = positive|\mathbf{e}) < 0.8$, and so $TL(Gate_1)$ is not satisfied and $Gate_1$ does not trigger. Variable EC_S belongs to an inactive BN, and thus will not supply any posterior to $Gate_2$ (according to Definition 5), and therefore $TL(Gate_2)$ will not be satisfied. At this point in time we have not observed any evidence for variables W and Z.
- Set 2: X and Y are updated as before. This time we will assume that $p(EC_B = positive|\mathbf{e}) > 0.8$, satisfying $TL(Gate_1)$ and triggering $Gate_1$. This will activate the BN $Sell$, and W will be instantiated according to the evidence. Assume $p(EC_S = negative|\mathbf{e}) < 0.6$, then $Gate_2$ does not trigger. According to Definition 6, all parent BNs of triggered gates that are not children of triggered gates are deactivated. This implies that Buy is deactivated. Feeding the triggered gates into the decision function results in a buy signal for the trader.
- Set 3: W and Z are updated according to the evidence as $Sell$ now is active. Since Buy now is inactive, evidence for X and Y is discarded. Assume that $TL(Gate_2)$ is not satisfied and due to Definition 5 we know that neither is $TL(Gate_1)$.

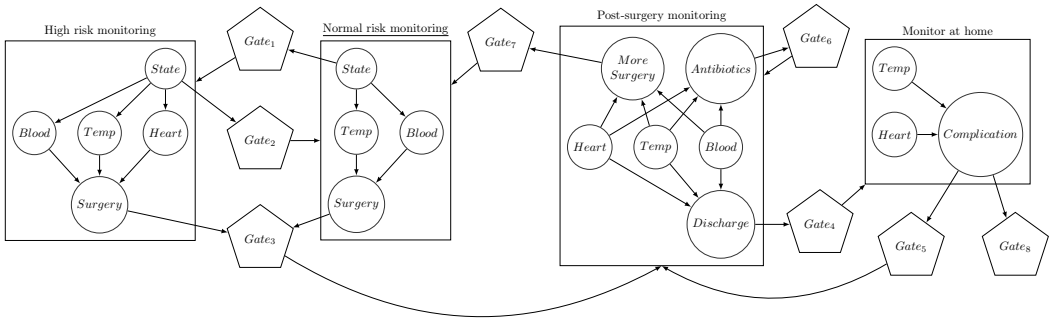

Figure 7: Surgery patient monitoring using a GBN

- Set 4: W and Z are updated as before. This time we will assume that $p(EC_S = negative|\mathbf{e}) > 0.6$, thus $TL(Gate_2)$ is satisfied and $Gate_2$ triggers. This will activate Buy, allowing A and B to be instantiated according to the new evidence. Assume that $p(EC_B = positive|\mathbf{e}) < 0.8$, then $Gate_1$ does not trigger. This leads to $Sell$ being deactivated. Feeding the triggered gates into the decision function results in a sell signal for the trader.

3.2 Patient Monitoring

In this section we will introduce an illustrative example of using a GBN in a different domain than algorithmic trading. Here the number of phases involved has increased, and not all gates are mapped to an explicit decision. The example in this section also sets the stage for the comparison to other models done in Section 6. The GBN in Figure 7 models a process relating to a particular patient prior to and after surgery. Equation 4 defines the decision function for this GBN.

$$Decision|triggered\ gates = \begin{cases} \text{Perform surgery if } Gate_3 \in triggered\ gates \\ \text{Discharge patient if } Gate_4 \in triggered\ gates \\ \text{Readmit patient if } Gate_5 \in triggered\ gates \\ \text{Give antibiotics if } Gate_6 \in triggered\ gates \\ \text{Perform surgery if } Gate_7 \in triggered\ gates \\ \text{Stop monitoring if } Gate_8 \in triggered\ gates \\ \text{Do nothing if none of the above apply} \end{cases} \tag{4}$$

In the BN $Normal\ risk\ monitoring$, we only measure the patients temperature ($Temp$) and blood pressure ($Blood$) to decide whether or not it is appropriate to perform surgery. At the same time we are classifying the patient as either being in a normal state or in a high risk state (using the variable $State$). If the posterior probability of being in a high risk state is above some threshold, then $Gate_1$ will trigger, thus activating the BN $High\ risk\ monitoring$ and deactivating $Normal\ risk\ monitoring$ (the switching model is a threshold model using posterior probabilities). When the patient is in the high risk state we also check the heart rate of the patient ($Heart$) to decide if it is time to perform surgery. Meanwhile, we are

monitoring the risk/normal state of the patient, and if the posterior probability of the patient being in the normal state is above some threshold then $Gate_2$ will trigger, thus switching back and forth between the two monitoring phases. Notice that the triggerings of $Gate_1$ and $Gate_2$ do not lead to any explicit decisions (however in this specific case it is implicitly necessary for somebody to add or remove the heart rate monitoring device, unless it is always connected but not used).

At any time the posterior probability of $Surgery = true$ can exceed the threshold of $TL(Gate_3)$, thus indicating that it is appropriate to perform surgery, triggering the gate and deactivating both $Normal\ risk\ monitoring$ and $High\ risk\ monitoring$, and activating the BN *Post-surgery monitoring*. In this example some of the networks are using the same variables and the decision stays the same (whether or not to perform surgery), however the conditional probabilities of the variables are different, and it could also be the case that the threshold is different in $Gate_3$ depending on which $Surgery$ variable is supplying the posterior, e.g. let $Surgery_{Normal}$ be the $Surgery$ variable in $Normal\ risk\ monitoring$ and $Surgery_{High}$ the variable in $High\ risk\ monitoring$, then it would be possible to define $TL(Gate_3) : p(Surgery_{Normal}|\mathbf{e}) > 0.6 \vee p(Surgery_{High}|\mathbf{e}) > 0.8$.

After surgery, in *Post-surgery monitoring*, three gates can trigger, each one associated with a decision. Either the trigger logic of $Gate_7$ is satisfied and the decision is made to have another round of surgery, thus coming back to the normal/high risk monitoring phases. If the trigger logic of $Gate_6$ is satisfied then a round of antibiotics is given to the patient, and the post-surgery monitoring continues. If the patient is deemed healthy enough to be discharged, then the trigger logic of $Gate_4$ will be satisfied, and the monitoring can continue at home using the BN $Monitor\ at\ home$. When the patient is at home the blood test has been removed, but the temperature and heart rate is still measured. In case there is a high posterior probability of complications at home, $Gate_5$ will trigger, thus sending the patient back to the post-surgery monitoring at the hospital.

If the patient is at home, and the posterior probability of a complication is very low, then $Gate_8$ will trigger, leading to the decision to stop monitoring the patient. The entire GBN comes to a halt, as $Gate_8$ has no child BNs and no more evidence is collected.

We will use the patient monitoring example to highlight some key differences to other models and formalisms in Section 6.

4 Learning Algorithm

Having defined GBNs and shown examples of their use, we turn our attention to the task of using GBNs in real-world applications. In order to do so we must somehow learn which GBN to use in a given situation. To this end, we will in this section introduce a semi-automatic algorithm for learning a GBN. The algorithm consists of two parts: a *GBN template* and a novel combination of k-fold cross-validation and time series cross-validation (time series cross-validation is sometimes known as *rolling origin* [14] or *walk forward analysis* [15]). We first describe these two parts, and then define the learning algorithm itself.

4.1 Gated Bayesian Network Templates

A GBN template is a representation of the modelled phases, including the possible transitions between them. The template defines where BNs and gates can be placed. For each slot where a BN can be placed, there is a library of BNs to choose from, similarly so for gates

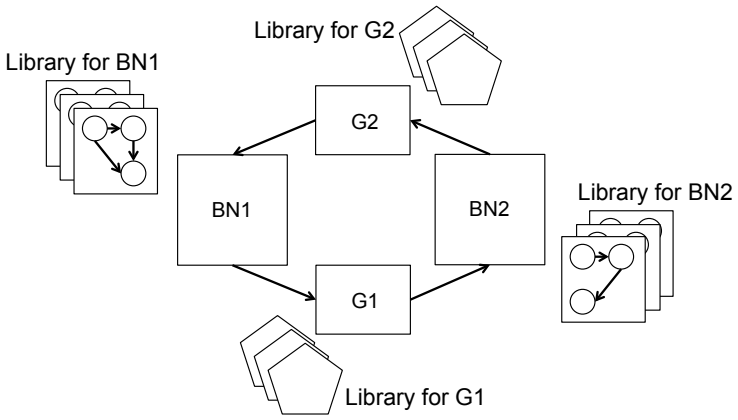

Figure 8: GBN template

(gates differ in their trigger logic, e.g. the thresholds may vary between them). BNs may be hand-crafted by experts prior to the GBN modelling, or they may be learnt from data using some structure learning algorithm. In any case, it is expected that the user provides the template and the libraries, hence this is why the algorithm is semi-automatic. A template with four slots and corresponding libraries is depicted in Figure 8.

Selecting a BN and a gate from the libraries for each slot in the template creates a GBN (e.g. Figure 1), we call this a candidate of the template. We use $\mathcal{C}_i$ to denote GBN candidate i of a GBN template. Since the structure of the BNs and the trigger logic of the gates in the libraries are defined, the remaining free parameters of a GBN candidate C_i are the parameters of the marginal and conditional probability distributions of the contained BNs, which we will denote by Θ.

The only restrictions on the BNs and gates are the ones they place on each other, e.g. if the trigger logic of the gates placed in G2 includes an expression about the posterior probability of a negative economical climate, then the BNs placed in BN2 must contain a node that can supply this value. Except for these restrictions, the BNs and gates can be configured freely.

4.2 Splitting the Data

When dealing with sequential data (in time or space) it is common that the data used to estimate the parameters of a given model always come before the data used to test the model. Future data may contain evolutionary effects of past data, and may therefore be more indicative of past data than past data is of future data. Thus, estimating the parameters Θ of the marginal and conditional probability distribution on future data and testing on past data, may be misleading if the goal is to evaluate the expected performance. A data set $\mathcal{D}$ of consecutive evidence sets, e.g. observations over all or some of the random variables in the GBN, is divided into n equally sized blocks $(\mathcal{D}_1, ..., \mathcal{D}_n)$ such that they are mutually exclusive and exhaustive. Each block contains consecutive evidence sets and all evidence sets in block $\mathcal{D}_i$ come before all evidence sets in $\mathcal{D}_j$ for all $i < j$.

Depending on the amount of available data, k is chosen as the number of blocks used for training. These blocks will be used to pick a promising candidate which should be evaluated on the testing data. In order to maximise the usage of the training data, we

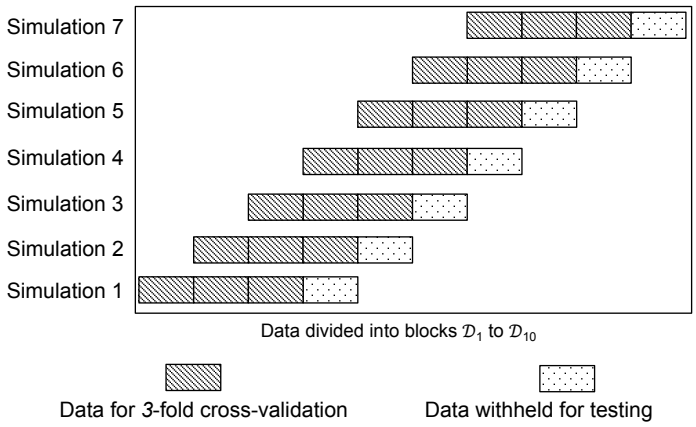

Figure 9: Combined k-fold cross-validation and time series cross-validation using $n = 10$ blocks and $k = 3$ folds

ignore the natural order of the data during training and use k-fold cross-validation. It should be noted that this is safe, since we only do this when choosing a promising candidate to evaluate, and do not use this scheme when evaluating the expected performance of the algorithm. Training then consists of holding out one of the k blocks (known as the validation data), and estimating the parameters Θ of the candidate using the rest of the blocks. This continues until every block in the training data has been held-out and validated upon.

Starting from index 1, blocks $1, .., k$ are used for training and $k + 1$ for testing, thus ensuring that the evidence sets in the testing data occurs after the training data (as in time series cross-validation). The procedure is then repeated starting from index 2 (i.e. blocks $2, .., k + 1$ are used for training and $k + 2$ for testing). By doing so we create repeated simulations, moving the testing data one block forward each time. An illustration of this procedure when $n = 10$ and $k = 3$ is show in Figure 9.

4.3 Algorithm

Let $\mathcal{J}$ be a score function such that $\mathcal{J}(\mathcal{C}_i, \mathcal{D}_j, \{\mathcal{D}_l, ..., \mathcal{D}_m\})$ is the score for GBN candidate C_i (which is candidate i of a GBN template, as defined in Section 4.1) when block j has been used for either testing or validation and the blocks $\mathcal{D}_l, ..., \mathcal{D}_m$ have been used to estimate the parameters Θ. The algorithm then works in three steps (with an optional fourth):

1. For each simulation s, where (as discussed previously) $\mathcal{D}_{s+k}$ is the testing data and $\mathcal{D}_s, ..., \mathcal{D}_{s+k-1}$ is the training data, find C^s that satisfies Equation 5. This corresponds to finding the GBN candidate with the maximum mean score of the k evaluations performed during k-fold cross-validation over the training data. This is done by taking into consideration every possible candidate, thus exhausting the search space.

$$\mathcal{C}^s = \arg\max_{\mathcal{C}_i} \ \frac{1}{k} \Sigma_{j=s}^{s+k-1} \mathcal{J}(\mathcal{C}_i, \mathcal{D}_j, \{\mathcal{D}_s, ..., \mathcal{D}^{s+k-1}\} \backslash \mathcal{D}_j) \ . \tag{5}$$

2. For each $\mathcal{C}^s$ calculate its score $\rho^s_{\mathcal{J}}$ on the testing set with respect to the scoring function $\mathcal{J}$ according to Equation 6. This corresponds to estimating Θ of the found GBN

candidate from Equation 5 using all training data and evaluating the performance on the data withheld for testing.

$$\rho^s_{\mathcal{J}} = \mathcal{J}(\mathcal{C}^s, \mathcal{D}_{s+k}, \{\mathcal{D}_s, ..., \mathcal{D}^{s+k-1}\}) \ . \tag{6}$$

3. The expected performance $\bar{\rho}_{\mathcal{J}}$ of the algorithm, with respect to the score function $\mathcal{J}$, is then given by the average of the scores $\rho^s_{\mathcal{J}}$, as described in Equation 7.

$$\bar{\rho}_{\mathcal{J}} = \frac{1}{n-k} \Sigma_{s=1}^{n-k} \rho^s_{\mathcal{J}} \ . \tag{7}$$

4. (Optional) If the objective is to find the candidate to be used on future unseen data (i.e. block $\mathcal{D}_{n+1}$) then Equation 5 is used once more to find $\mathcal{C}^{n-k+1}$. This candidate can then be used on $\mathcal{D}_{n+1}$ with an expected performance $\bar{\rho}_{\mathcal{J}}$.

We emphasise that to ensure that each candidate is assessed on several blocks before selecting the one to move forward with, we allow that a validation block may come before some blocks used for parameter estimation in Equation 5. However, this step is only used to select a candidate to evaluate in Equation 6. The data used in Equation 6 is always ordered, i.e. the test block is always the immediate successor of the blocks used for parameter estimation.

In the description of the algorithm, one scoring function $\mathcal{J}$ has been used both for choosing a promising candidate in Equation 5 and for evaluating the expected performance of the algorithm in Equation 6. In Section 5.1.2 we will define several metrics used to evaluate algorithmic trading systems. The scoring function $\mathcal{J}$ used in Equation 5 could internally use many of these metrics to come up with one score to compare the different candidates with. However, it is natural in the coming setting to expose the actual values of these metrics in Equation 6, and so several scoring functions $\mathcal{J}$ can be used to get a vector of scores $[\rho^s_{\mathcal{J}_1}, ..., \rho^s_{\mathcal{J}_m}]$ and use a vector of means as the performance of the algorithm $[\bar{\rho}_{\mathcal{J}_1}, ..., \bar{\rho}_{\mathcal{J}_m}]$.

5 Application

We can now address our initial motivation for introducing GBNs, to use them as alpha models in algorithmic trading systems (defined in Section 1.1). We aim to use our learning algorithm to learn a GBN as an alpha model that generates buy and sell signals, such that certain risks (that will be defined in Section 5.1.2) are mitigated as compared to the buy-and-hold strategy, while at the same time maintaining similar or better rewards. In this section, we first introduce several metrics that are used to evaluate the performance of alpha models, and then we move to the experiment itself.

5.1 Evaluating Alpha Models

Regression models can be evaluated by how well they minimise some error function or by their log predictive scores. For classification, the accuracy and precision of a model may be of greatest interest. Alpha models may rely on regression and classification, but cannot be evaluated as either. For an alpha model, it is not important to accurately predict every movement of the market, but rather to identify events in the market that suggest

an opportune time to buy or sell. Therefore, optimising alpha models by using classical supervised classification measures such as accuracy, precision, recall, etc. will not be in line with the desired behaviour of the model. To clarify, it is not necessarily known prior to learning when these opportune times are, and so the task is in this sense unsupervised, as there is no way of guiding the model to which events it should classify correctly. An alpha model's performance needs to be based on its generated signals over a period of time, and the performance must be measured by the risk and reward of the model. This is known as *backtesting*.

5.1.1 Backtesting

The process of evaluating an alpha model on historic data is known as backtesting, and its goal is to produce metrics that describe the behaviour of a specific alpha model. These metrics can then be used for comparison between alpha models [15, 16]. A time range, price data for assets traded and a set of signals are used as input. The backtester steps through the time range and executes signals that are associated with the current time (using the supplied price data) and computes an *equity curve* (which will be explained in Section 5.1.2). From the equity curve it is possible to compute metrics of risk and reward. To simulate potential transaction costs, often referred to as *commission*, every trade executed is usually charged a small percentage of the total value (0.06% is a common commission charge used in the included application).

Alpha models are backtested separately from the other components of the algorithmic trading system, as the backtesting results are input to the other components. Therefore, we execute every signal from an alpha model during backtesting, whereas in a full algorithmic trading system we would have a portfolio construction model that would combine several alpha models and decide how to build a portfolio from their signals.

5.1.2 Alpha Model Metrics

What constitutes risk and reward is not necessarily the same for every investor, and investors may have their own personal preferences. However, there are a few metrics that are common and often taken into consideration [16]. Here we will introduce the metrics that we will use to evaluate the performance of our alpha models.

Although not a metric on its own, the *equity curve* needs to be defined in order to define the following metrics. The equity curve represents the total value of a trading account at a given point in time. If a daily timescale is used, then it is created by plotting the value of the trading account day by day. If no assets are bought, then the equity curve will be flat at the same level as the initial investment. If assets are bought that increase in value, then the equity curve will rise. If the assets are sold at this higher value then the equity curve will again go flat at this new level. The equity curve summarises the value of the trading account including cash holdings and the value of all assets. We will use $\mathcal{E}_t$ to reference the value of the equity curve at point t.

Metric 1 (Return) The *return* of an investment is defined as the percentage difference between two points on the equity curve. If the timescale of the equity curve is daily, then $r_t = (\mathcal{E}_t - \mathcal{E}_{t-1})/|\mathcal{E}_{t-1}|$ would be the *daily* return between day t and $t-1$. We will use $\bar{r}$ and σ_r to denote the mean and standard deviation of a set of returns.

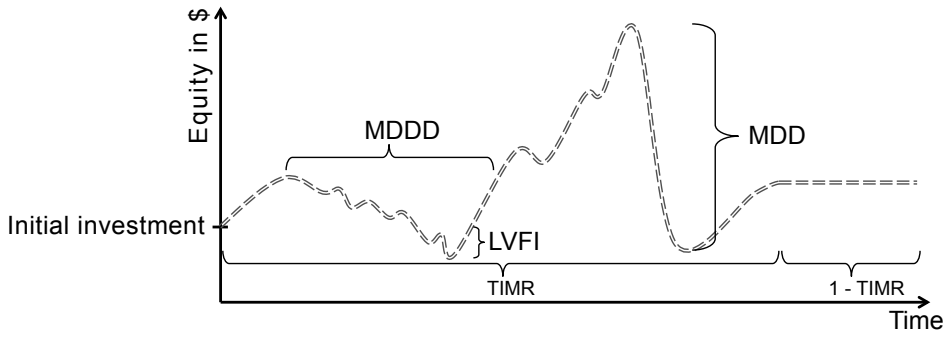

Figure 10: Example of an equity curve with drawdown risks

Metric 2 (Sharpe Ratio) One of the most well known metrics used is the so called *Sharpe ratio*. Named after its inventor Nobel laureate William F. Sharpe, this ratio is defined as: $(\bar{r} - \text{risk free rate})/\sigma_r$. The *risk free rate* is usually set to be a "safe" investment such as government bonds or the current interest rate, but is also sometimes removed from the equation [16]. The intuition behind the Sharpe ratio is that one would prefer a model that gives consistent returns (returns around the mean), rather than one that fluctuates. This is important since investors tend to trade *on margin* (borrowing money to take larger positions), and it is then more important to get consistent returns than returns that sometimes are large and sometimes small. This is why the Sharpe ratio is used as a reward metric rather than the return.

Furthermore, under certain assumptions it can be shown that there exists an optimal allocation of equity between alpha models (in the portfolio construction model), such that the long-term growth rate of equity is maximised [16]. This growth rate turns out to be $g = r + S^2/2$, where r is the risk free rate and S is the Sharpe ratio. Thus, a high Sharpe ratio is not only an indication of good risk adjusted return, but holding the risk free rate constant, the optimal growth rate is an increasing function of the Sharpe ratio.

Using the Sharpe ratio as a metric will ensure that the alpha models are evaluated on their risk adjusted return, however, there are other important alpha model behaviours that need to be measured. A family of these, that are known as *drawdown risks*, are presented here (see Figure 10 for examples of an equity curve and these metrics).

Metric 3 (Maximum Drawdown (MDD)) The percentage between the highest peak and the lowest trough of the equity curve during backtesting. The peak must come before the trough in time. The MDD is important from both a technical and psychological regard. It can be seen as a measure of the maximum risk that the investment will live through. Investors that use their existing investments that have gained in value as safety for new investments may be put in a situation where they are forced to sell everything. Other risk management models may automatically sell investments that are loosing value sharply. For the individual who is not actively trading but rather placing money in a fund, the MDD is psychologically frustrating to the point where the individual may withdraw their investment at a loss in fear of loosing more money.

Metric 4 (Maximum Drawdown Duration (MDDD)) The longest it has taken from one peak of the equity curve to recover to the same value as that peak. Despite its unfor-

tunate name it is not the duration of the MDD, but rather then longest drawdown period. There is an old adage amongst investors to "cut your losses early". In essence it means that it is better to take a loss straight away than to sit on an investments for months or years, hoping that it will come back to positive returns. During this time one could have reinvested the money elsewhere, rather then breaking-even much later (or taking a larger loss much later). Models that have long periods of drawdown lock resources when they could have been used better elsewhere.

Metric 5 (Lowest Value From Investment (LVFI)) The percentage between the initial investment and the lowest value of the equity curve. This is one of the most important metrics, and has a significant impact on technical and psychological factors. For investors trading on margin, a high LVFI will cause the lender to ask the investor for more safety capital (known as a *margin call*). This can be potentially devastating, as the investor may not have the capital required, and is then forced to sell the investment. The investor will then never enjoy the return the investment could have produced. Individuals who are not investing actively, but instead are choosing between funds that invest in their place, should be aware of the LVFI as it is the worst case scenario if they need to retract their investment prematurely.

Metric 6 (Time In Market Ratio (TIMR)) The percentage of time of the investment period where the alpha model owned assets. This metric may seem odd to place within the same family as the other drawdown risks, however it fits naturally in this space. We can assume that the days the alpha model does not own any assets the drawdown risk is zero. If we are not invested, then there is no risk of loss. In fact, we can further assume that our equity is growing according to the risk free rate, as it is not bound in assets.

5.1.3 Buy and Hold Benchmark

At first the buy-and-hold strategy may seem naïve, however it has been shown that deciding when to own and not own assets requires consistent high accuracy of predictions in order to gain higher returns than the buy-and-hold strategy [17]. The buy-and-hold strategy has become a standard benchmark, not only because of the required accuracy, but also because it requires very little effort to execute (no complex computations and/or experts needed). In the current setting we are dealing with buy and sell signals of single assets, however in a wider context where several assets are considered at the same time, portfolio construction creates a stark contrast to the the buy-and-hold strategy. Portfolio construction actively reallocates resources between different assets, while buy-and-hold never reallocates. However, theoretical results show that in the long run the universal portfolio [9], and other online portfolio construction algorithms [8, 10], outperform the buy-and-hold strategy under certain criteria. The Markowitz portfolio [7] is another example that emphasises that diversification and reallocation can improve expected rewards while reducing risk.

Now consider the family of metrics that we called drawdown risks. The buy-and-hold strategy holds assets over the entire backtesting period and so will be subject to the full force of these metrics. For instance, as an asset will be held throughout the period, the lowest point of the assets value will coincide with LVFI. Furthermore, the initial investment will always be locked in assets, not being able to make money from risk free rates during periods of decreasing value.

Table 1: Calculation of indicators

S is a set of ordered values. n is the number of periods used.	
Moving average	$MA_t(S,n) = \frac{1}{n}\sum_{i=0}^{n-1} S_{t-i}$
Moving average difference	$MADIFF_t(S, n_{fast}, n_{slow}) = \frac{MA_t(S,n_{fast}) - MA_t(S,n_{slow})}{MA_t(S,n_{slow})}$
Relative strength index	$Up_t(S,n) = \{\lvert S_i - S_{i-1}\rvert : S_i > S_{i-1} \text{ with } t-n < i \leq t\}$ $Down_t(S,n) = \{\lvert S_i - S_{i-1}\rvert : S_i < S_{i-1} \text{ with } t-n < i \leq t\}$ $RS_t(S,n) = \frac{Up_t(S,n)}{Down_t(S,n)}$ $RSI_t(S,n) = 100 - \frac{100}{1+RS_t(S,n)}$

5.2 Methodology

The variables used in the BNs of our GBNs were all based on so called technical analysis. One of the major tenets in technical analysis is that the movement of the price of an asset repeats itself in recognisable patterns. *Indicators* are computations of price and volume that support the identification and confirmation of patterns used for forecasting [18, 19, 20]. Many classical indicators exists, such as the moving average (MA), which is the average price over time, and the relative strength index (RSI) which compares the size of recent gains to the size of recent losses. Technical analysis is a topic that is being actively developed and researched [21, 22]. In this application we used three indicators: the MA, the RSI and the relative difference between two MAs (MADIFF). Please see Table 1 for the calculations of these indicators.

5.2.1 GBN Template

A GBN template with one BN per phase was created (see Figure 8), along with eight BNs per BN slot (see Figure 11) and four gates per gate slot, giving a total of 1024 candidates. The eight BNs used for BN1 were identical to those used in BN2, however the gates' trigger logic were different. The trigger logic for G1 asks for the posterior probability of a good buying opportunity (i.e. a predicted positive future climate) while the trigger logic for G2 asks for the posterior probability of a good selling opportunity (i.e. a predicted negative future climate). Each one of the four gates available for G1 and G2 had different thresholds which the posterior probability had to exceed in order for the gate to trigger (the thresholds were 0.5, 0.6, 0.7 and 0.8). The choice of variables and the structure of the eight BNs represent different experts' views on how to interpret the technical analysis indicators. For instance, in Figure 11 network 2 represents an expert that believes that RSI measured at its current value and its value five days in the past are indicative of future price movements, while network 3 believes the same but using the difference between two moving averages. These views are not exhaustive, however we assumed that these were the experts available at the time of the application.

The random variables in the BNs were discretisations of technical analysis indicators (RSI, MA and MADIFF) and their corresponding first and second order 1 and 5 day backward finite differences ($\nabla_1^1, \nabla_5^1, \nabla_1^2$ and ∇_5^2) which approximate the first and second order derivatives. The parameters used in the indicators are standard 14 day period for RSI [18]

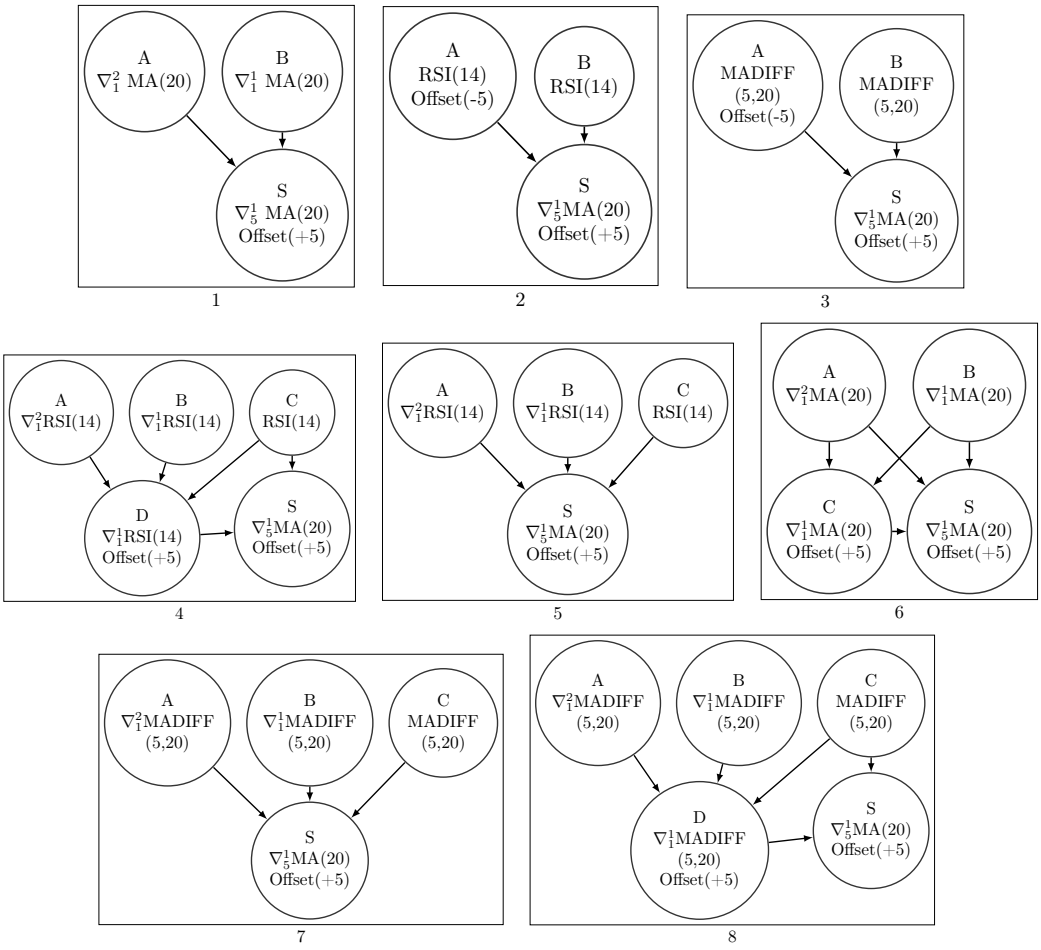

Figure 11: BNs in GBN template libraries

(written as RSI(14)), 20 day period for MA, representing 20 trading days in a month (written as MA(20)), and 5 and 20 day period for MADIFF, where 5 days represent the 5 trading days in a week (written as MADIFF(5,20)). We also considered the previous indicators but with an offset of 5 days in the past and 5 days into the future. The random variables that were offset into the future represent the future economical climate, one of which was involved in the trigger logic of the gates. The true values for these future random variables were naturally not part of the testing data sets. The nodes named S in Figure 11 were used as trigger nodes for all gates. The GBN generated trading signals as it transitioned between its two phases.

5.2.2 Data Sets

A set of actively traded stock shares were chosen for the evaluation of our learning algorithm: Apple Inc. (AAPL), Amazon.com Inc. (AMZN), International Business Machines Corporation (IBM), Microsoft Corporation (MSFT), NVIDIA Corporation (NVDA), Gen-

eral Electric Company (GE), Red Hat Inc. (RHT). The daily adjusted closing prices for these stocks between 2003-01-01 and 2012-12-31 were downloaded from Yahoo! Finance^{TM}. This gave a total of 10 years of price data for each stock, where each year was allocated to a block, and thus n = 10. For the learning algorithm, k was chosen to be 3, giving seven simulations from which to calculate $[\bar{\rho}_{\mathcal{J}_1}, ..., \bar{\rho}_{\mathcal{J}_m}]$. The split of the data is visualised in Figure 9.

5.2.3 Scoring Functions

The signals generated were backtested in order to calculate the relevant metrics. For step 1 in the learning algorithm (see Section 4.3) we used the Sharpe ratio. This choice was made as it combines both risk and reward into one score, which can then easily be compared between candidates. For step 2 we used the return and drawdown risks described in Section 5.1.2 to create a score vector. For the buy-and-hold strategy the same metrics as in step 2 were calculated for the seven simulations.

5.3 Results and Discussion

To visualise the backtesting that was done for each simulation, Figure 12 gives two examples of stock price, generated signals (an upward arrow indicates a buy signal and a downward arrow indicates a sell signal) and resulting equity curve (with an initial investment of $20,000 USD) for the evaluated GBN. The solid line equity curve is the one achieved by executing the signals from the GBN, the dashed line is the corresponding equity curve for the buy-and-hold strategy. The GBN equity curve grows in a more monotonic fashion, which is desirable because this decreases the drawdown risks, while at the same time generating positive returns. The buy-and-hold strategy would have made a loss in both these examples, because the final price is lower than the initial one, furthermore it would have displayed bad intermediate behaviour, reflected by the high drawdown risk values that would have been incurred. These are declining years for the shares, however the GBN does its best to get as much value as possible from the price movements.

Table 2 presents the score vectors from the learning algorithm versus the score vector of the buy-and-hold strategy over the seven simulations. Rows named *min*, *max* and *sd* (standard deviation) are based on Equation 6, while *mean* corresponds to Equation 7. As each block used by the learning algorithm had an approximate length of one year, the Sharpe ratio that is given by dividing the mean with the sd of the return column is a yearly Sharpe ratio based on seven years (where the risk-free rate has not been included). The acronyms MDD, MDDD, LVFI and TIMR, represent the metrics described in Section 5.1.2. All values are ratios except for MDDD which is measured in number of days.

5.3.1 Analysis of Results

The Sharpe ratio is our measure of reward, premiered above the raw return for reasons discussed in Section 5.1.2. Our first concern is to ensure that the learnt GBNs are producing similar or better Sharpe ratios than the buy-and-hold strategy over the testing period. As can be seen in Table 2, this is the case except for NVDA and RHT. As we have previously discussed, it requires a very high accuracy of predictions to consistently beat the Sharpe ratio of buy-and-hold.

From this we can conclude that the GBNs do not get beaten consistently by the buy-and-hold strategy when considering the annual Sharpe ratio, even though it is considered a

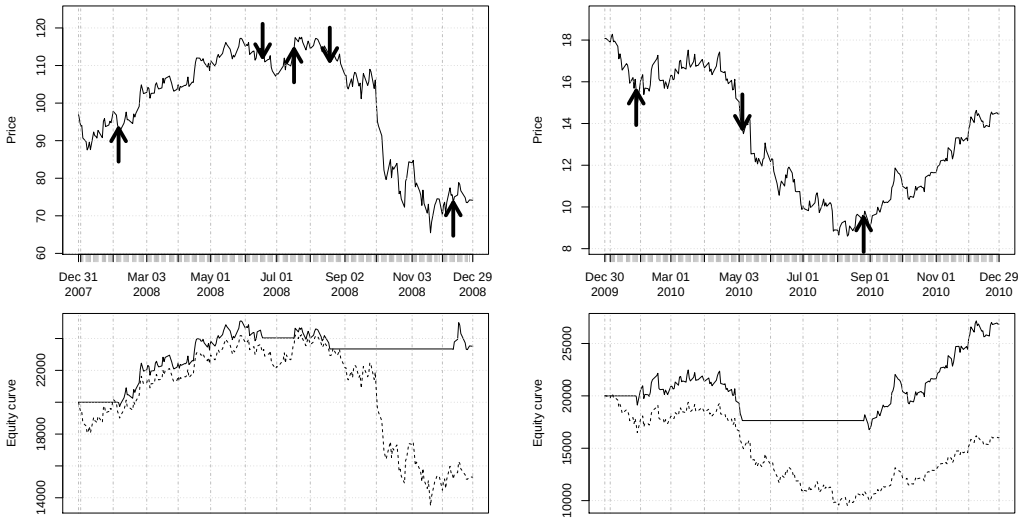

Figure 12: Price, signals and GBN equity curve for IBM 2008 (left) and NVDA 2010 (right)

Table 2: Metric values comparing GBN with buy-and-hold

		GBN					Buy-and-hold				
		Return	MDD	MDDD	LVFI	TIMR	Return	MDD	MDDD	LVFI	TIMR
AAPL	min	-0.000	0.122	35.0	0.001	0.520	-0.559	0.129	28.0	0.001	1.000
	max	0.851	0.331	164.0	0.184	0.944	1.419	0.589	250.0	0.590	1.000
	mean	0.347	**0.206**	**95.0**	**0.055**	**0.723**	0.489	0.274	116.0	0.162	1.000
	sd	0.334	0.076	50.3	0.061	0.155	0.707	0.168	82.7	0.218	0.000
	Sharpe	**1.041**					0.691				
AMZN	min	-0.204	0.134	56.0	0.042	0.510	-0.466	0.157	45.0	0.001	1.000
	max	0.784	0.306	142.0	0.245	0.768	1.740	0.634	249.0	0.620	1.000
	mean	0.271	**0.218**	**101.7**	**0.109**	**0.630**	0.463	0.317	118.6	0.215	1.000
	sd	0.374	0.060	32.8	0.088	0.091	0.829	0.171	89.9	0.234	0.000
	Sharpe	**0.725**					0.559				
IBM	min	-0.022	0.062	53.0	0.013	0.494	-0.210	0.088	28.0	0.001	1.000
	max	0.238	0.176	176.0	0.121	0.944	0.596	0.442	190.0	0.302	1.000
	mean	0.125	**0.117**	112.3	**0.044**	**0.712**	0.170	0.174	**106.4**	0.086	1.000
	sd	0.094	0.042	45.4	0.042	0.173	0.245	0.120	59.7	0.101	0.000
	Sharpe	**1.332**					0.694				
MSFT	min	-0.256	0.099	88.0	0.001	0.365	-0.457	0.141	74.0	0.001	1.000
	max	0.381	0.305	197.0	0.279	0.741	0.659	0.498	250.0	0.498	1.000
	mean	0.056	**0.168**	**143.3**	**0.114**	**0.557**	0.069	0.249	168.6	0.200	1.000
	sd	0.202	0.068	41.9	0.091	0.156	0.338	0.119	67.8	0.155	0.000
	Sharpe	**0.278**					0.204				
NVDA	min	-0.420	0.182	64.0	0.032	0.241	-0.765	0.253	67.0	0.077	1.000
	max	0.342	0.541	227.0	0.467	0.700	1.230	0.820	249.0	0.821	1.000
	mean	0.016	**0.284**	**148.1**	**0.209**	**0.516**	0.202	0.458	172.3	0.311	1.000
	sd	0.284	0.120	62.1	0.140	0.171	0.701	0.195	76.6	0.268	0.000
	Sharpe	0.057					**0.288**				
GE	min	-0.302	0.049	60.0	0.015	0.404	-0.555	0.089	69.0	0.001	1.000
	max	0.461	0.465	217.0	0.438	0.570	0.222	0.657	217.00	0.642	1.000
	mean	0.040	**0.169**	**144.3**	**0.119**	**0.488**	-0.001	0.314	157.0	0.236	1.000
	sd	0.235	0.142	69.7	0.150	0.062	0.257	0.228	53.7	0.257	0.000
	Sharpe	**0.169**					-0.005				
RHT	min	-0.222	0.096	87.0	0.001	0.433	-0.370	0.143	40.0	0.001	1.000
	max	0.436	0.428	221.0	0.348	0.784	1.341	0.676	221.0	0.617	1.000
	mean	0.038	**0.254**	156.9	**0.136**	**0.613**	0.201	0.338	**133.0**	0.243	1.000
	sd	0.259	0.103	45.6	0.123	0.136	0.579	0.197	61.6	0.234	0.000
	Sharpe	0.145					**0.346**				

nearly optimal strategy. Furthermore, we should take into consideration TIMR. The GBNs are spending less time in the market, reducing risk to equity and possibly increasing equity value from risk free investments. Potential gain in equity from risk free rates have not been added to the Sharpe ratios presented in the table. Considering that the learnt GBNs consistently spend considerably less time in the market (shown by the low TIMR values), this could give a significant boost to the Sharpe ratios. An example of this can be seen for NVDA where the Sharpe ratio for GBN is lower than for buy-and-hold, but the GBN only spent on average 51.6% of the time in the market, risk free investments could potentially drive the Sharpe ratio for the GBN above that of the buy-and-hold strategy.

Turning our attention to the drawdown risks, we first consider the MDD and MDDD. The difference of the MDD values are substantial, the MDD mean and sd are consistently smaller for the GBNs than they are for the buy-and-hold strategy. This signals that the equity we gain from our investments are at less risk when using the GBNs compared to the buy-and-hold strategy. For MDDD the means differ in favour of either approach, we would not prefer one in front of the other given only this metric.

The LVFI is a major threat to equity (see Section 5.1.2), and it is the one metric where buy-and-hold severely underperforms. Considering the max values we note that for NVDA the buy-and-hold strategy wiped out 82.1% of the equity at worst, while the GBNs did 46.7% at worst for NVDA. Considering the LVFI mean and sd for all stocks we note that they are consistently almost half for the GBNs compared to the buy-and-hold strategy. LVFI is important because it is the risk of the initial investment, loosing much of the initial investment may lead to premature withdrawal of funds and/or force liquidation by margin-calls.

All in all, the results above clearly indicate that GBNs are competitive with buy-and-hold in terms of Sharpe ratio, whereas they induce a more desirable behaviour in terms of MDD, LVFI and TIMR.

5.3.2 Single Bayesian Network Comparison

We have made a leap into immediately assuming that having different BNs for the different phases in Figure 8 would be an improvement above having the same BN in both phases. This assumption stems from the underlying hypothesis of GBNs, that different BNs are required at different phases of a process. However, in order to illuminate upon the difference between using the same BN for each phase compared to our results in Table 2, we ran the same experiment again, however this time only using the subset of candidate GBNs that had the same BN in both phases. For brevity we only report the comparison of annual Sharpe ratios in Table 3. As is evident, GBNs with different BNs in the different phases outperform the single BN for all stocks. For some stocks the difference is marginal (NVDA, GE and RHT), while for others the difference is substantial (AAPL, AMZN, IBM and MSFT).

From this we conclude that there is evidence for the underlying hypothesis of GBNs, that different BNs are required at different phases of a process. Without having further investigated the origins of this improvement, it does seem to suggest that buying opportunities and selling opportunities are not each others counterparts.

Table 3: Annual Sharpe ratio for single BN and GBN

	AAPL	AMZN	IBM	MSFT	NVDA	GE	RHT
Single BN	0.675	0.561	0.44	0.0181	0.0432	0.142	0.12
GBN	**1.041**	**0.725**	**1.332**	**0.278**	**0.057**	**0.169**	**0.145**

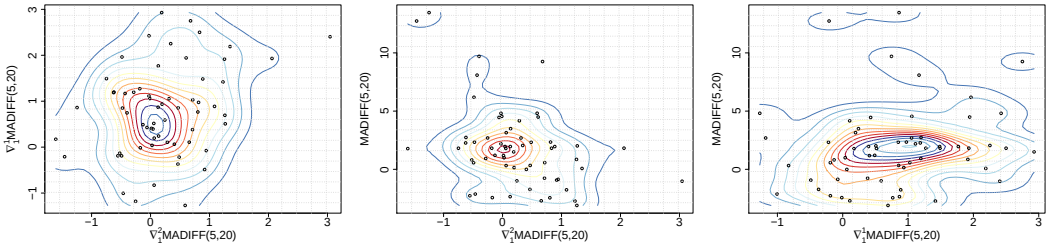

Figure 13: Buy decisions using 7 from Figure 11

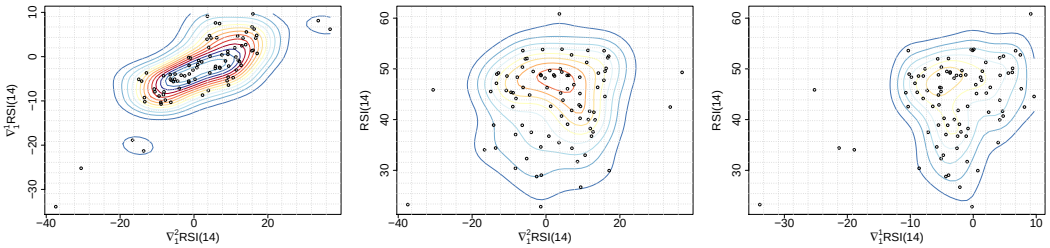

Figure 14: Sell decisions using 5 from Figure 11

5.3.3 Post-Analysis

One of the benefits of using BNs is that we can get transparency as to why a particular signal was generated. Our aim here was to look at the non-discretised values of the variables at the time a signal was generated. We combined the signals from all simulations (regardless of which stock was traded) and then grouped the signals by which BN generated them and if they were buy or sell signals. We then did pair-wise combinations of the variables in each BN to create scatter plots with values of the variables along the axes and also added an approximated density using the frequency of signals. These scatter plots show when GBNs are generating signals. Examples of these plots for the BNs that generated the most signals are given in Figure 13 (using 7 from Figure 11) and Figure 14 (using 5 from Figure 11).

In Figure 13 the BN is used to look for buying opportunities. In the first plot we see that most signals are generated when both $\nabla_1^1 MADIFF(5,20)$ and $\nabla_1^2 MADIFF(5,20)$ are positive, indicating that the difference between the two MAs is growing and increasing in speed, but not so positive so as to making it impossible to benefit from the trend. The second two plots in Figure 13 plot $\nabla_1^2 MADIFF(5,20)$ against $MADIFF(5,20)$ and the $\nabla_1^1 MADIFF(5,20)$ against $MADIFF(5,20)$. Both these confirm what we knew about the first and second order difference, but also indicate that $MADIFF(5,20)$ should be positive (so the short period MA should be above the long period MA). From a technical analysis perspective this type of pattern is common, it indicates a trend change, as the shorter MA is moving above and away from the longer MA. It is noteworthy to mention that we have not set any priors on the BNs that would indicate that these are the kind of patterns we are interested in, so our learning algorithm is able to re-discover these human-like commonly used patterns. An example of selling signals is presented in Figure 14, here we are using RSI which is bounded between 0 and 100. When RSI moves up towards 100 it indicates that the buying pressure is increasing, and should drive prices higher, the opposite is true when

RSI moves towards 0. The first plot indicates that most selling signals are generated when $\nabla_1^1 RSI(14)$ is close to zero or negative (i.e. RSI has started to decrease) and $\nabla_1^2 RSI(14)$ is bounded around ± 10. The two other plots in Figure 14 represent $\nabla_1^2 RSI(14)$ against $RSI(14)$ and $\nabla_1^1 RSI(14)$ against $RSI(14)$. These last two figures confirm our findings in the first figure, but also indicates that the $RSI(14)$ should be below 50 (but not too much below 50 so as to miss the selling opportunity). This seems reasonable from a technical analysis perspective, as RSI goes below 50 and decreases, the selling pressure increases, indicating that the price will go lower, and so a selling signal is generated. We reemphasise that we did not set any prior in the BNs that would suggest that these are the type of signals we should be looking for.

5.4 Extended Experiment

In the main application we used the posterior probability of a variable being in a specific state as input to the trigger logic of the gates. Signals generated were due to the posterior probability being higher or lower than some threshold, and we used the Sharpe ratio to pick a candidate after k-fold cross-validation to use on the testing data. In this way, we did not directly communicate to the candidates that they should premier a specific score, but instead picked the candidate that already behaved as desired. Since we did not beat the Sharpe ratio of buy-and-hold for NVDA and RHT, we wanted to attempt to improve these results. According to Definition 1 and Definition 5 we can introduce utility nodes into the BNs and use them as trigger nodes. By doing so, we aimed at instructing the candidates to behave in such a way that they premiered the Sharpe ratio.

The Sharpe ratio itself is defined as the mean of a set of returns divided by the standard deviation of those returns (see Section 5.1.2). Therefore, it is calculated after the returns have been realised, and is therefore not an expectation. At the time the trigger logic for a gate is evaluated, the information available is the posterior weighted utilities given some evidence. Thus, it is not possible to compute the Sharpe ratio for a specific set of evidence, but rather the expected risk adjusted return. Given a discrete variable S that represents the future return of an investment over some period of time, we can construct a logical statement regarding the expected risk adjusted return according to Equation 8. This is a reasonable approach for instructing the candidates to premier the Sharpe ratio.

$$
\begin{aligned}
TL(\cdot) : \frac{\mu}{\sqrt{\sum_i (u(S=i)-\mu)^2 p(S=i|\mathbf{e})}} &> \tau \\
\mu = \sum_i p(S=i|\mathbf{e}) u(S=i) & \\
p(S=i|\mathbf{e}) \text{ probability that S is in state } i \text{ given evidence } \mathbf{e} & \\
u(S=i) \text{ utility of S when it is in state } i &
\end{aligned}
\tag{8}
$$

All trigger nodes (S in Figure 11) represent the first order 5 day backward finite difference of the 20 day moving average of the share price, offset 5 days into the future. These trigger nodes were replaced with the 5 day *return* of the 20 day moving average, offset 5 days into the future. We kept the 20 day moving average as a smoother in order to make as few changes as possible between the two experiments. These new nodes were discretised into six bins, and a utility node was added that mapped each one of these bins to the mean value of the content of the bin (the candidates looked similar to Figure 2). The trigger logic

for the gates were set according to Equation 8. This implies that each time a GBN was presented with new evidence, it calculated an expected risk adjusted return based on the current posterior distribution of the variable S, and if this was higher or lower than some threshold τ then it triggered, generating a signal.

The results of the extended experiment are shown in Table 4. For compactness we only present the mean values of each score and the annual Sharpe ratio. For AAPL and AMZN we did not improve upon the results from the main application. For IBM, MSFT, NVDA, GE and RHT we were able to improve the Sharpe ratio. For NVDA the improvement was clear, but not enough to beat buy-and-hold, on the other hand for RHT we were able to improve the Sharpe ratio enough to beat buy-and-hold. It does however seem to come at a cost, as the other score metrics were not necessarily improved upon over all stocks. This is most likely due to the fact that we prioritised the Sharpe ratio by using the expected risk adjusted return in the trigger logic.

Table 4: Metric values comparing GBN, GBN with utility and buy-and-hold

		Sharpe	MDD	MDDD	LVFI	TIMR
AAPL	GBN	**1.041**	**0.206**	**95.0**	**0.055**	**0.723**
	GBN with utility	0.821	0.236	105	0.0857	0.807
	Buy-and-Hold	0.691	0.274	116.0	0.162	1.000
AMZN	GBN	**0.725**	**0.218**	**101.7**	**0.109**	**0.630**
	GBN with utility	0.469	0.292	117	0.183	0.692
	Buy-and-Hold	0.559	0.317	118.6	0.215	1.000
IBM	GBN	1.332	0.117	112.3	**0.044**	**0.712**
	GBN with utility	**1.44**	**0.112**	108	0.0536	0.713
	Buy-and-Hold	0.694	0.174	**106.4**	0.086	1.000
MSFT	GBN	0.278	0.168	143.3	0.114	**0.557**
	GBN with utility	**0.376**	**0.162**	**143**	**0.0989**	0.572
	Buy-and-Hold	0.204	0.249	168.6	0.200	1.000
NVDA	GBN	0.057	**0.284**	**148.1**	**0.209**	**0.516**
	GBN with utility	0.216	0.313	149	0.22	0.518
	Buy-and-Hold	**0.288**	0.458	172.3	0.311	1.000
GE	GBN	0.169	0.169	144.3	0.119	**0.488**
	GBN with utility	**0.416**	**0.129**	**137**	**0.0716**	0.51
	Buy-and-Hold	-0.005	0.314	157.0	0.236	1.000
RHT	GBN	0.145	**0.254**	156.9	**0.136**	**0.613**
	GBN with utility	**0.485**	0.257	153	0.153	0.734
	Buy-and-Hold	0.346	0.338	**133.0**	0.243	1.000

6 Comparison to Other Models

In the previous sections we have formally defined GBNs as well as introduced algorithms that can be used when applying GBNs to real-world problems. Our intention for doing so is to complement the set of existing probabilistic graphical models with GBNs. In this section we will compare GBNs with existing models and formalisms, in order to highlight a few key differences.

6.1 Influence Diagrams and Markov Decision Processes

When the decisions, actions and possible outcomes can be made explicit in the model, then we usually consider decision models. Influence diagrams (IDs) are, within the domain of probabilistic graphical models, the canonical models for dealing with decisions under uncertainty. An example of an ID is drawn in Figure 15. We express decisions and possible actions in a given scenario using decision nodes (square nodes) and express the value of each

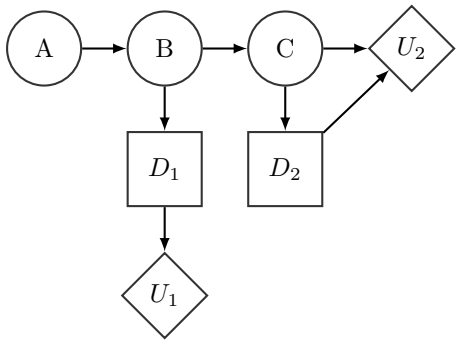

Figure 15: An example of an influence diagram

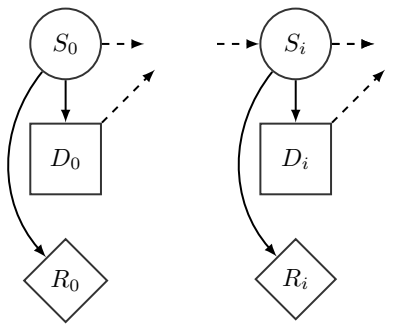

Figure 16: An example of a Markov decision process

possible outcome using utility nodes. Given a fully specified ID and a set of evidence, we are able to extract the actions for each decision such that the expected utility is maximised [2]. This is referred to as finding the optimal strategy given the current evidence. This strategy can be computed due to the fact that all decisions and possible outcomes are explicitly modelled. If a decision needs to be repeated several times, it is possible to unfold the influence diagram, much like a dynamic Bayesian network. It is then assumed that each decision is separated by a discrete time step that does not change in length, allowing chance and decision nodes to be connected between the time steps. Limiting the unfolding is the fact that it is necessary to explicitly bound the number of decisions in the process. Several obstacles when using IDs have been overcome, specifically concerning asymmetry in decision processes [23, 24].

In situations where it is necessary to model a decision process with an unbounded number of decisions, it is possible to use a Markov decision processes (MDP), an example of which is shown in Figure 16. In the framework of MDPs, it is more common to phrase the problem as deciding to move between states, given the current state and evidence, and that each decision changes the state with some probability. Being in a specific state is associated with some reward R (that can also be a cost if negative). Using *value iteration*, we can find a utility function that maps all states to a decision such that it maximises the expected utility. However, as the time horizon is not bounded, summing up utilities will lead to an infinite sum, and so in order for value iteration to work one must apply constraints on the model. A common way of doing this is to discount future rewards, so that rewards in the distant future fades away. Value iteration finds the utility of a state by looking at the immediate reward received at that specific state plus the maximum expected utility of all future states that are reachable from that state [2]. Thus, if at time j you can decide to move to a state where you own shares $s_{j+1} = \textit{own shares}$ or a state where you do not own shares $s_{j+1} = \textit{do not own}$, then the utility function will map the current state to a decision of which state to move to, based on the maximum expected utility of all future states. However, the MDP and value iteration assumes that the reward R of a state is independent of time j. This assumption becomes problematic when the reward depends on when and if we have made a decision at any previous time $i < j$. For instance, the value of our shares at time j will be different if we bought them at time i or time k ($i \neq k$). This would require dynamically changing the edges in the MDP depending on previous decisions.

In a GBN we do not explicitly model decisions and outcomes, and hence GBNs are

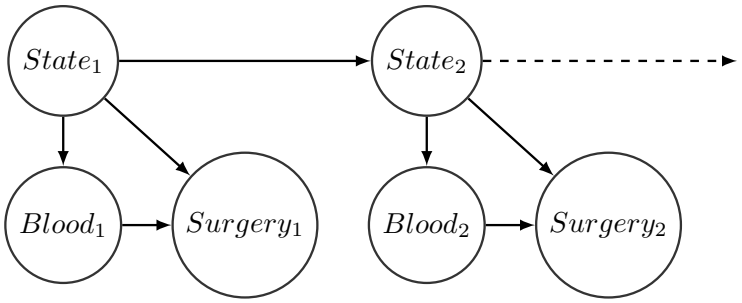

Figure 17: An example of a hidden Markov model

not decision models and do not solve the maximum expected utility problem. Instead, the learning algorithm in Section 4 finds the candidate which induces the best strategy Δ with respect to some score function $\mathcal{J}$. Put in other words, learning a GBN entails finding the candidate that encodes the strategy under which switching of BNs occur in such a way that the best score over a sequence of evidence sets is achieved. It is clear that this is different from what IDs and MDPs achieve. GBNs repeat decisions similar to when folding out IDs, however it severs the ties between time slices, i.e. there are no edges between different BNs in a GBN, and can thus handle evidence at variable time steps (e.g. there is no need to specify that decisions are separated by one hour or one day, etc.). Furthermore, GBNs handle unbounded time horizons as MDPs do, however they do not use a discounting scheme or value iteration. There is no reward function R used in the trigger logic, thus a gate that triggers does not do so due to expected utility maximisation of future states, but due to the strategy the GBN encodes.

6.2 Hidden Markov Models

In the example discussed in Section 3.2, we used a GBN to switch between two models based on the posterior probability of a patient being in a specific unobservable state. This is closely related to the way hidden Markov models (HMM) are used. However, there are a few fundamental differences that we will explore.

Assume that we have some data $\mathcal{D}$ which contains cases that we can estimate our parameters from. In $\mathcal{D}$, the true state of the patient is known, however at the time of the decision, we cannot know the true state of the patient but must instead infer it. We can split the data into $\mathcal{D}_{normal}$ with cases where the patient state is at normal risk and $\mathcal{D}_{high}$ where the patient state is at high risk. For the GBN in Figure 7, we estimate the parameters of $Normal\ risk\ monitoring$ using $\mathcal{D}_{normal}$ and $High\ risk\ monitoring$ using $\mathcal{D}_{high}$. The variables $Blood$, $Temp$ and $Heart$ are always observed, thus separating $State$ from $Surgery$, however as the data has been split, $Surgery$ in $Normal\ risk\ monitoring$ is implicitly conditioned on $State$ in $Normal\ risk\ monitoring$, and so an edge between them is not necessary.

The HMM in Figure 17 is a simplified version of the scenario discussed. For this model, we would estimate from $\mathcal{D}$ the parameters of all the distributions implied by the model. We have to add an edge between $State_t$ and $Surgery_t$ as $Blood_t$ is always observed and it would separate $State_t$ from $Surgery_t$.

We have assumed that the patients state is not observable at the time of decision, hence

the posterior of $Surgery_t$ will be influenced of all other variables in the HMM, for all time steps. The HMM effectively becomes a mixture model, where the posterior of $Surgery_t$ is weighed by the likelihood of each state at each previous time step. Now assume that we could observe the state of the patient at $t = 2$. The posterior of $Surgery_2$ is now only influenced by $State_2$ and $Blood_2$.

In the GBN, we do not estimate what is known as the transition model $p(State_{t+1}|State_t)$, thus we do not specify any conditional probability of activating and deactivating BNs. This conditional inherently exists due to the trigger logic and the threshold, however it is never explicitly specified. Furthermore, although $State$ is not observable, the GBN makes an active choice based on the strategy Δ; a GBN will deterministically select a network based on the posterior belief that a patient is in a specific state and some threshold, there is no mixture between the two networks. Thus the posterior probability of $Surgery$ is not a mixture of all variables of all previous time steps. This is intentional, as GBNs were introduced in order to deal with systems of distinct phases.

The differences become even more discernible when considering the trader's problem, where we do not want the posterior of negative climate to become a mixture of all previous time steps, as some of the time steps should have been modelling positive climate. This ties into the fact that in the GBN it is straightforward to add and remove variables and dependencies based on which phase we are in. This could potentially be done in the HMM as well, however it would require some artificial augmenting of the nodes, edges and distributions, and as one of the main reasons of using a graphical model is to use it as a tool for communication, we prefer the explicit expressiveness of GBNs in this case.

6.3 Context Specific Independence

In the example in Section 3.2, some variables are sometimes independent of each other and sometimes not, depending on the evidence that has been supplied. This is similar to what is known as context specific independence [25], where previously dependent variables may become independent given a set of evidence (i.e. a context). The DAG representing a BN does not reveal these asymmetries, instead the conditional probability distributions must be estimated and investigated in order to identify them. This has been viewed as a shortcoming of BNs, and therefore extensive work has been done to represent the context specific independencies without having to investigate the probability distribution parameters. Chain event graphs (CEGs) were introduced as a probabilistic graphical model that represents these asymmetries explicitly, while at the same time being able to represent the BN in case of symmetry [26]. CEGs are defined as a function of an event tree, which represents how a certain process may unfold, and certain symmetries in this event tree. The symmetries are exploited to create a graph where steps in the process that are the same are collapsed. This not only decreases the size of the event tree, but also makes explicit the asymmetric relationships among steps in the process. CEGs have since further been developed in order to model processes with non-finite event trees [27].

While GBNs also are capable of explicitly describing some of the asymmetries that context specific independence entails, it is quite different from those explained by [25] and [26]. Since GBNs use BNs to model the phases, asymmetries can still exist in these BNs that are not made explicit in the graph. GBNs make explicit asymmetries that occur once a process has changed phase, i.e. once a set of evidence has fulfilled the conditions of the trigger logic of some gate. These conditions are not necessarily fulfilled by one single variable taking one specific state, but is a joint condition on potentially several variables and the

threshold of the trigger logic. Therefore, it is more accurate to say that GBNs make explicit context specific independencies that are due to the context established by a deterministic latent variable, for which the states are determined by the strategy Δ that the GBN encodes (as was described in Section 2.2).

Another distinct difference between CEGs and GBNs is the fact that CEGs encode a probability of moving from one step to another in the event tree. GBNs do not encode this probability, i.e. it is not made explicit that with probability α the GBN in Figure 1 will transition from the *Buy* phase to the *Sell* phase. Instead, transitions in a GBN depend on the posterior probabilities inferred by the contained BNs over the observations made.

6.4 Other Formalisms

Taking a large BN and dividing it into smaller subnets has been researched for some time. Notable contributions include multiply sectioned Bayesian networks [28, 29], agent encapsulated Bayesian networks [30], and object-oriented Bayesian networks [31]. Although these frameworks all section the BNs into smaller parts, they still come together, unlike GBNs, to create one large BN. Also, GBNs are sensitive to the order of the evidence supplied. Similar evidence sensitivity can be found in research regarding structural adaptation [32] and query based diagnostics [33]. These two approaches add or remove nodes and edges from the graphical model to adapt to the changing data, while GBNs do not handle the entry of new variables or dependencies into the model.

7 Conclusions and Future Work

Based upon the need to represent a process that goes back and forth between distinct phases, we have introduced GBNs as a new probabilistic graphical model to complement the existing set of such models. In this paper we have offered a formal definition of GBNs, as well as an algorithm to be used to execute the GBN given a set of data. Furthermore, we have introduced a learning algorithm to semi-automatically learn the structure of the GBN, and shown how it can be applied in the domain of algorithmic trading.

7.1 Learning Algorithm

The semi-automatic learning algorithm proposed in Section 4 does require the availability of a set of BNs to populate the libraries. The algorithm presented does not explicitly define how this should be done. Depending on the context, a combination of experts and structure learning algorithms may be helpful in designing these BNs. However, in our ongoing research we are investigating the idea of incorporating structure learning and feature selection as part of the overall learning algorithm for GBNs. Intuitively, this should improve upon our results as we can explore a broader spectrum of relationships between the dynamics of the technical analysis indicators and their predictive power of future price movements.

Although incorporating structure learning and feature selection as part of the GBN learning algorithm may yield better results, a more pressing matter is the fact that exhaustive evaluation of every possible GBN candidate from a GBN template is only feasible when the number of BNs and gates are small. Running our experiments on a 3.2 GHz Intel Core i5 only took six minutes per asset, allowing us to keep matters clear and not involve any heuristics for candidate selection in the learning algorithm. However we do acknowledge the fact that as the model grows, it will be necessary to prune the search space considerably.

One way of doing this, which we are currently exploring, is the use of Bayesian optimisation [34]. Succinctly, Bayesian optimisation makes assumptions regarding the smoothness of a score function that should be maximised, and by only testing points where the score function's value has high uncertainty, the number of tests can be reduced significantly. In our first experiments we are fixing the structure of the BNs, but allowing the parameterisation of the technical indicators to be optimised by Bayesian optimisation, changing the number of available BNs to hundreds of thousands rather than eight, while keeping similar execution times [35] .

7.2 Application

The results show that learnt GBNs consistently reduce risk with similar or better rewards, compared to the benchmark buy-and-hold, and do so while at the same time staying out of the market for considerable amounts of time. During these non-invested days the equity is at zero risk and can gain value from risk free assets. We were also able to improve the Sharpe ratio, for some of the stocks, by introducing utility nodes as trigger nodes, thus being able to instruct the candidates to prioritise the Sharpe ratio. As was mentioned in Section 5.1.2, under certain assumptions the long-term mean growth rate is optimised by optimising the Sharpe ratio, however as these assumptions may not always hold, taking into consideration drawdown risks is still very necessary. Given this fact, and the results of our extended experiment, it may be useful to adopt a multi-criteria scoring mechanism, such as a weighted mean-rank, both in the trigger logic and when picking the winning candidate in the learning algorithm.

7.3 Other Models and Formalisms

The benefit of using a GBN over a single BN was illustrated in Section 5.3.2, and evidence was found to support the underlying hypothesis of GBNs. We also discussed how GBNs are different from other existing frameworks, specifically how GBNs do not solve the expected utility maximisation problem, but rather encode a strategy of how to handle a series of evidence, where the strategy is optimised with respect to some score. We have also explained how GBNs deterministically switch between models, rather than specifying a mixture of models. Furthermore, we contrasted GBNs against CEGs that explicitly model conditional independence asymmetries, something that BNs are not capable of doing, and how GBNs do not make explicit the same types of asymmetries as CEGs . As a final note, although not discussed previously, we have very preliminary ideas on using GBNs to give explanations to models induced by chain graphs and vice versa [36].

Acknowledgments

The second author is funded by the Center for Industrial Information Technology (CENIIT) and a so-called career contract at Linköping University, and by the Swedish Research Council (ref. 2010-4808).

References

[1] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann Publishers, 1988.

[2] F. V. Jensen and T. D. Nielsen, *Bayesian networks and decision graphs*. Springer, 2007.

[3] K. B. Korb and A. E. Nicholson, *Bayesian artificial intelligence*. Taylor and Francis Group, 2011.

[4] P. Treleaven, M. Galas, and V. Lalchand, "Algorithmic trading review," *Communications of the ACM*, vol. 56, no. 11, pp. 76–85, 2013.

[5] G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree, "Algorithmic trading," *Computer*, vol. 44, no. 11, pp. 61–69, 2011.

[6] R. K. Narang, *Inside the black box*. John Wiley & Sons, 2013.

[7] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[8] B. Li and C. H. Hoi, "Online portfolio selection: A survey," *ACM Computing Surveys*, vol. 46, no. 3, pp. 1–36, 2014.

[9] T. M. Cover, "Universal portfolios," *Mathematical Finance*, vol. 1, no. 1, pp. 1–29, 1991.

[10] D. P. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth, "On-line portfolio selection using multiplicative updates," *Mathematical Finance*, vol. 8, no. 4, pp. 325–347, 1998.

[11] M. Bendtsen and J. M. Peña, "Gated Bayesian networks," in *Proceedings of the Twelfth Scandinavian Conference on Artificial Intelligence*, pp. 35–44, 2013.

[12] M. Bendtsen and J. M. Peña, "Learning gated Bayesian networks for algorithmic trading," in *Proceedings of the Seventh European Workshop, Probabilistic Graphical Models*, pp. 49–64, 2014.

[13] C. Bielza and P. Larrañaga, "Discrete Bayesian network classifiers: A survey," *ACM Computing Surveys*, vol. 47, no. 3, pp. 1–43, 2014.

[14] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: an analysis and review," *International Journal of Forecasting*, vol. 16, no. 4, pp. 437–450, 2000.

[15] R. Pardo, *The evaluation and optimization of trading strategies*. John Wiley & Sons, 2008.

[16] E. P. Chan, *Quantitative trading*. John Wiley & Sons, 2009.

[17] W. F. Sharpe, "Likely gains from market timing," *Financial Analysts Journal*, vol. 31, no. 2, pp. 60–69, 1975.

[18] J. J. Murphy, *Technical analysis of the financial markets*. New York Institute of Finance, 1999.

[19] M. J. Pring, *Technical analysis explained*. McGraw-Hill, 2002.

[20] R. D. Edwards and J. Magee, *Technical analysis of stock trends*. Martino Publishing, 1957.

[21] "Journal of technical analysis." www.mta.org, first published in 1978.

[22] "International federation of technical analysts journal." www.ifta.org/publications, first published in 2000.

[23] F. V. Jensen and M. Vomlelová, "Unconstrained influence diagrams," in *Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 234–241, 2002.

[24] F. V. Jensen, T. D. Nielsen, and P. P. Shenoy, "Sequential influence diagrams: a unified asymmetry framework," *International Journal of Approximate Reasoning*, vol. 42, no. 1–2, pp. 101–118, 2006.

[25] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller, "Context-specific independence in Bayesian networks," in *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 115–123, 1996.

[26] J. Q. Smith and P. E. Anderson, "Conditional independence and chain event graphs," *Artificial Intelligence*, vol. 172, no. 1, pp. 42–68, 2008.

[27] L. M. Barclay, R. A. Collazo, J. Q. Smith, P. A. Thwaites, and A. E. Nicholson, "The dynamic chain event graph," *Electronic Journal of Statistics*, vol. 9, no. 2, pp. 2130–2169, 2015.

[28] Y. Xiang and V. Lesser, "Justifying multiply sectioned Bayesian networks," in *Proceedings of the Fourth International Conference on Multi-Agent Systems*, pp. 349–356, 2000.

[29] Y. Xiang, "Multiply sectioned Bayesian networks and junction forests for large knowledge-based systems," *Computational Intelligence*, vol. 9, no. 2, pp. 171–220, 1993.

[30] S. Langevin, M. Valtorta, and M. Bloemeke, "Agent-encapsulated Bayesian networks and the rumor problem," in *Proceedings of the Nineth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1553–1554, 2010.

[31] D. Koller and A. Pfeffer, "Object-oriented Bayesian networks," in *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 302–313, 1997.

[32] S. H. Nielsen and T. D. Nielsen, "Adapting Bayes network structures to non-stationary domains," *International Journal of Approximate Reasoning*, vol. 49, no. 2, pp. 379–397, 2008.

[33] J. M. Agosta, T. R. Gardos, and M. J. Druzdzel, "Query-based diagnostics," in *Proceedings of the Fourth European Workshop, Probabilistic Graphical Models*, pp. 1–8, 2008.

[34] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," Tech. Rep. UBC TR-2009-023 and arXiv:1012.2599, 2009.

[35] M. Bendtsen, "Bayesian optimisation of gated Bayesian networks for algorithmic trading," *Bayesian Modeling Application Workshop (BMAW)*, 2015, to appear.

[36] J. M. Peña, "Every LWF and AMP chain graph originates from a set of causal models," in *Proceedings of the 13th European Conference on Symbolic and Quantitative Approaches to Reasoning under Uncertainty (ECSQARU 2015) – Lecture Notes in Artificial Intelligence*, 2015, to appear.