3

RELATED WORK

3.1 INTRODUCTION

Influence relations, in observational data, are mainly deduced though statistical methods [Hatfield *et al.* 2006; Opgen-Rhein and Strimmer 2007; Grinthal and Berkeley Heights 2015; Commenges and Gégout-Petit 2009]. Discovering influence between stochastic processes using statistical methods has been explored with regard to Bayesian network structure learning between random variables which is of great importance to many sciences [Bunge 2017; Salmon 1984; Koller and Friedman 2009].

We may not always be given the structure of a Bayesian network in advance, and might want to learn the Bayesian network structure for either knowledge discovery or density estimation [Heckerman *et al.* 1995a; Mohammadi and Wit 2015; Madsen *et al.* 2017; Fan *et al.* 2014]. On the one hand, if we learn fewer edges than those in the true network structure, there is no possible setting of the parameters that will give us the joint distribution specified in the true network [Koller and Friedman 2009]. On the other hand, if we learn more edges than those in the true network, we might be able to capture the true distribution, but risk fragmentation given more edges than necessary [Koller and Friedman 2009]. Considering the pressures that fragmentation imposes on parameter estimation [D'Elia *et al.* 2003], it might be better to generalise the true distribution with a sparser structure even though we might not completely capture the true distribution [Koller and Friedman 2009]. There are two common approaches to structure learning in Bayesian networks: constraint-based structure learning [Colombo and Maathuis 2014] and score-based structure learning [Koller and Friedman 2009; Campos and Ji 2011].

In *constraint-based* structure learning [Campos and Ji 2011] we view the Bayesian network structure as a set of independence assumptions, for which we can discover a perfect map by performing multiple hypothesis tests for conditional independence between variables [Lehmann and Romano 2006; Cheng *et al.* 1997]. Statistical tests often can be wrong about certain independence assertions [Kanji 2006]. When learning the distribution of influence between processes we are required to test a large number of hypotheses (given the length of the processes involved). If we are wrong about certain hypotheses, this error could grow which reduces our ability to recover the true Bayesian network structure regardless of the tests significance level [Koller and Friedman 2009]. Thus declaring independence assertions which are incorrect will bias the learning procedure to an incorrect structure [Koller and Friedman 2009] which reduces our ability to recover the true set of independence assumptions. This is since every choice of adding an independence assumption restricts the type of assumptions which can be added in future. Therefore, constrant-based approaches is best suited to networks with a small amount of variables; a large amount of samples with

strong presence of dependencies; and is efficient to finding the true structure when faced with a structure which as already quite close to it [Koller and Friedman 2009]. Constrant-based approaches are therefore unsuitable for our task of tracking influence between partially observed multi-dim processes. Another promising approach, called score-based structure learning [Bouchaala *et al.* 2010], resolves these restrictions by considering the complete structure as a state in the search space.

Score-based structure learning requires the definition of a hypothesis space of potential network structures; defines a scoring function which gages how well the model fits the observed training data; and finally, attempts to find the highest scoring network structure as an optimisation problem [Kok and Domingos 2005; Tenenbaum *et al.* 2011; Tsamardinos *et al.* 2006; Lee *et al.* 2007]. However, given that the search space is super-exponential in size (given the number of possible DAGs with respect to the number of variables considered), this proposes an NP-hard problem which we attempt to solve using heuristic techniques [Chickering *et al.* 1994; Chickering 1996; Chickering *et al.* 2004; Suzuki 2017].

Score-based structure learning is best suited to the task of learning influence networks since it (a) considers the complete influence structure as a state in the search space (Section 3.6); (b) preserves basic score properties to allow for feasible computations (Section 3.2 and Section 3.7); and (c) it provides a clear indication of the independence assertions between the concerned networks relative to the data (Section 3.6) [Koller and Friedman 2009; Campos and Ji 2011; Ellis and Wong 2008]. In this section we review related traditional score-based structure learning practices.

This chapter is structured as follows. In Section 3.2 we explore a score which provides the likelihood of a proposed structure relative to the data. In Section 3.3 we introduce an extension to the likelihood score called the Bayesian information criterion (BIC) which makes use of a penalty to manage structural complexity. In Section 3.4 we discuss the Bayesian score which uses the Bayesian paradigm. In Section 3.5 we review related structure search techniques to recover tree structures. In Section 3.6 we present practices that recover graph structures which is a much more difficult problem than recovering tree structures. Finally, in Section 3.7 we present the complexity of the general structure learning task.

3.2 THE LIKELIHOOD SCORE

Recall that score-based structure learning requires the definition of a scoring function which gages how well the model fits the training data. There are several choices of scoring functions geared at evaluating the likelihood of a particular structure given the fit to data [Drton and Maathuis 2017]. A well-known choice is that of the likelihood score which maximises the likelihood (or log-likelihood in practice) of the structure to the data [Beretta *et al.* 2017]. We can express this using the maximum likelihood estimate, $\hat{\theta}$, given a particular graph structure, $\mathcal{G}$, relative to the data, $\mathcal{D}$. Thus we denote the likelihood score more formally as,

$$\text{score}_L = \ell((\hat{\theta}, \mathcal{G}) : \mathcal{D}). \tag{7}$$

In other words, if we are presented with a particular graph structure we find the MLE of that graph with respect to the data. This can be expressed more generally as the relative cost of adding an edge between two variables in an empty graph structure. More formally, we can define the likelihood score as follows:

Proposition 3.1. The likelihood score decomposes as the number of instances multiplied by the mutual information, $\mathbf{I}_{\hat{P}}$, between each family of variables minus the entropy of each variable that is independent of the structure. More formally,

$$\text{score}_L(\mathcal{G}, \mathcal{D}) = M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}_{X_i}^{\mathcal{G}}) - M \sum_{i=1}^{n} \mathbf{H}_{\hat{P}}(X_i),$$

where

$$\mathbf{I}_{\hat{P}}(X; Y) = \sum_{x,y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)},$$

and

$$\mathbf{H}_{\hat{P}}(X) = -\sum_{x} P(x) \log P(x).$$

Intuitively, the mutual information informs us of the average cost of adding an edge between X and Y (ie. $P(x, y)$), over modelling these as mutually independent (ie. $P(x)P(y)$) [Koller and Friedman 2009]. Since the entropy term is independent of the graph structure, it is often ignored.

Proposition 3.1 intuitively tells us that the score will be higher if there is evidence in $\mathcal{D}$ which supports a high correlation of variables with their respective parents [Drton and Maathuis 2017; Liu *et al.* 1996]. However, the likelihood score has a strenuous consequence: maximising the likelihood score will always prefer the most connected network given that more edges always give more information about correlations between variables [Lee *et al.* 2007; Koller and Friedman 2009]. This is true unless variables are truly empirically independent, which is never the case in most datasets given statistical noise [Koller and Friedman 2009].

The fact that the most complicated network is always preferred poses a significant over-fitting problem. This is usually overcome by regularising the hypothesis space or penalizing structural complexity [Koller and Friedman 2009; Campos and Ji 2011]. In the next section we discuss a structure score which addresses this problem.

3.3 THE BAYESIAN INFORMATION CRITERION

The Bayesian information criterion (BIC), developed by Schwarz [1978], is a popular choice for trading off model complexity and fit to data. The BIC score consists of two terms: the first describes the fit of the hypothesised structure to the data, usually the likelihood function in Equation 7; and the second is a penalty term for complex networks. More formally the BIC score is given as

$$\text{score}_{BIC} = \ell(\hat{\theta}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2} \text{DIM}[\mathcal{G}], \tag{8}$$

where M is the number of training instances and the DIM[$\mathcal{G}$] is the number of independent parameters in the network. We note that the negation of this score is referred to as the minimal description length (MDL) score [Bouckaert 1993; Lam and Bacchus 1993; Suzuki 1993; Kullback 1997; Chow and Liu 1968].

Upon further investigation, we note that the entropy component (ie. $\mathbf{H}_{\hat{P}}(X)$) of the likelihood term in Equation 8 is negligible since it does not depend on the selected structure. This observation allows us to rewrite the BIC score as

$$\text{score}_{\text{BIC}} = M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}_{X_i}^{\mathcal{G}}) - \frac{\log M}{2} \text{DIM}[\mathcal{G}]. \tag{9}$$

As we increase the number of samples (ie. M) in Equation 9 the emphasis moves from model complexity to the fit to data [Chen and Gopalakrishnan 1998]. In other words, as we obtain more data we are more likely to consider more complicated structures [Tamura *et al.* 1991]. This property is referred to as consistency. More formally [Koller and Friedman 2009],

Definition 3.2. A scoring function is said to be *consistent* if, as the amount of data provided increases to infinity, the true structure will maximise the score; and all structures that are not I-equivalent (Definition 2.6) to the true structure will have a strictly lower score.

Alongside the definition of consistency (Definition 3.2) is the notion of score equivalence which states that I-equivalent structures have the same structure score. More formally,

Definition 3.3. A scoring function is said to be *score equivalent* if for all I-equivalent networks $\mathcal{G}$ and $\mathcal{G}'$ we have that $\text{score}(\mathcal{G} : \mathcal{D}) = \text{score}(\mathcal{G}' : \mathcal{D})$ for all datasets $\mathcal{D}$.

Schwarz [1978]; Rissanen [1987]; Barron *et al.* [1998] provided much development to the basic properties of the BIC score in addition to establishing its consistency and score equivalence which are used for efficient and manageable searches [Geiger *et al.* 2001; Rusakov and Geiger 2005; Settimi and Smith 1998]. In the next section we explore the Bayesian score which considers the Bayesian paradigm.

3.4 THE BAYESIAN SCORE

The last score that we discuss in detail is the Bayesian score. The Bayesian score, with a Dirichlet prior, was introduced by Buntine [1991]; Cooper and Herskovits [1992]; Spiegelhalter *et al.* [1993]; Heckerman *et al.* [1995a]. The Bayesian score operates under the Bayesian paradigm which states that anything we have uncertainty over we must maintain a probability distribution over. In this case we have uncertainty over parameters as well as network structure. More specifically,

$$P(\mathcal{G}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{G})P(\mathcal{G})}{P(\mathcal{D})}, \tag{10}$$

where $P(\mathcal{D}|\mathcal{G})$ is the marginal likelihood (not the maximum likelihood); $P(\mathcal{G})$ is the prior over structures; and finally, $P(\mathcal{D})$ is a constant term which describes the prior

over datasets. $P(\mathcal{D})$ gives us no information about the structure of the graph and can be ignored when being used to compare different structures. We can rewrite Equation 10 using a sum of logs as

$$\text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D}|\mathcal{G}) + \log P(\mathcal{G}). \tag{11}$$

The marginal likelihood term expands as follows

$$P(\mathcal{D}|\mathcal{G}) = \int_{\Theta_\mathcal{G}} P(\mathcal{D}|\theta_\mathcal{G}, \mathcal{G}) P(\theta_\mathcal{G}|\mathcal{G}) d\theta_\mathcal{G},$$

where $P(\mathcal{D}|\theta_\mathcal{G}, \mathcal{G})$ is the likelihood for data given the structure and parameters; and $P(\theta_\mathcal{G}|\mathcal{G})$ is the prior over the parameters. We note that the marginal likelihood integrates over all possible settings of the parameters (not just the maximum likelihood $\hat{\theta}$). Therefore in this case $\theta_\mathcal{G}$ is a setting of the parameters for a structure $\mathcal{G}$.

The idea of using the marginal likelihood, as an alternative to the maximum likelihood, significantly reduces over-fitting since it makes a prediction over an unseen instance given the previous instances (this is opposed to maximum likelihood which considers all the instances together).

Although the Bayesian score might seem complicated, when considering multinomial table CPDs the marginal likelihood can be provided in closed form as

$$P(\mathcal{D}|\mathcal{G}) = \prod_i \prod_{\mathbf{u}_i \in Val(Pa_{X_i}^\mathcal{G})} \frac{\Gamma(\alpha_{X_i|\mathbf{u}_i}^\mathcal{G})}{\Gamma(\alpha_{X_i|\mathbf{u}_i}^\mathcal{G} + M[\mathbf{u}_i])} \prod_{x_i^j \in Val(X_i)} \frac{\Gamma(\alpha_{x_i^j|\mathbf{u}_i}^\mathcal{G} + M[x_i^j, \mathbf{u}_i])}{\Gamma(\alpha_{x_i^j|\mathbf{u}_i}^\mathcal{G})},$$

where the gamma function is given by

$$\Gamma(x) = x.\Gamma(x-1).$$

It is important to recognise that the above expression allows us to express the log marginal likelihood as a sum over Bayesian family scores. Turning our attention to the structure prior in Equation 11, the most common choice is a uniform structure prior. Other structure priors include priors that are non-uniform over Bayesian network structures [Buntine 1991] or deviations between prospective networks and recommended networks [Heckerman *et al.* 1995b]. Another score closely related to the Bayesian score is the BDe score. The BDe score was proposed by Heckerman *et al.* [1995a], and the authors also showed that it possesses basic properties such as score-equivalence and local parameter independence.

In summary, the Bayesian score avoids over fitting by averaging over all possible network parameterisations and is asymptotically equivalent to the BIC score [Koller and Friedman 2009]. There has been thorough empirical analyses of the BIC and Bayesian scores [Dawid 1984; Kass and Raftery 1995]. Other common structure scores include the AIC score [Akaike 1974] which only considers the dimension of the graph structure (ie. the number of independent parameters) as a penalty term.

3.5 LEARNING TREE-STRUCTURED NETWORKS

Learning a tree structured network is perhaps the simplest structure learning problem. The first application of learning tree structured networks was proposed by Chow and Liu [1968]. Having selected a structure score, we turn our attention to an optimisation problem which attempts to maximise the selected score over potential structures. Decomposability of the score turns out to be an important property for computational savings in the structure search procedure. More specifically, decomposability is the property that the complete network score can be written as equal to the sum of family scores. For example, in Preposition 3.1 we see that the likelihood score decomposes as a sum of family scores (for every variables and its parent-set).

There are two important reasons why one would want to learn a tree structured network over a graph structured network. Firstly, there already exists powerful algorithms for efficient optimisation over high-dimensional tree structured networks; and secondly, trees provide sparse networks with manageable generalised parameterisation which reduces over-fitting. In summary, trees can be constructed by using polynomial time algorithms which summarise the most important dependencies that allow for better approximations [Koller and Friedman 2009; Bunge 2017]. Trees can also offer a suitable prior which can be used to learn more descriptive graph structures relative to the data [Koller and Friedman 2009].

In order to learn tree structures we need to calculate the score between different variables. If our score is decomposable we can express the weight between a variable, j, and its parent, i, as

$$w_{i\to j} = \text{score}(X_j|X_i) - \text{score}(X_j). \tag{12}$$

In the case of the likelihood score the expression of $w_{i\to j}$ in Equation 12 becomes

$$w_{i\to j} = M \sum_{x_i,x_j} P(x_i,x_j) \log \frac{P(x_i,x_j)}{P(x_i)P(x_j)}. \tag{13}$$

Intuitively, the weight between two variables is the average over the relative error of modelling the joint distribution between the variables and the product of them being marginally independent.

Now that we have a way to measure the score of two variables we can define an algorithm to obtain a tree structured network. A general algorithm to obtain a tree structured network is to compute the score of every pair of variables and then find the maximum weighted spanning tree (MWST) or forest between all of the variables. One could use any standard MWST algorithm such as Prim or Kruskal in $O(n^2)$ [Greenberg 1998; Huang *et al.* 2009], where n is the number of variables.

There are two important observations to note from Equation 13. The first is that the mutual information term can never be negative. However, in the BIC or BDe scores we could have a penalty term which causes $w_{i\to j}$ to be negative. If we use the likelihood score for $w_{i\to j}$ then the score between the two variables will be positive. Thus using the likelihood score with a MWST algorithm will result in a tree structured network as opposed to obtaining a forest when using penalty-based scores such as the AIC or BIC scores.

Secondly, score equivalent networks have the same score for $w_{i \to j}$ and $w_{j \to i}$, this implies that we will get a undirected tree or forest structure which we can then impose an orientation on the edges. One can use an arbitrary orientation on the edges, as long as the resulting graph is a DAG. In the next section we turn our attention to the much more difficult problem of learning graph structured networks.

3.6 LEARNING GENERAL GRAPH-STRUCTURED NETWORKS

In the case of learning general graph-structures the problem complexity increases greatly. More specifically [Koller and Friedman 2009],

Theorem 3.4. *For any dataset, $\mathcal{D}$, and decomposable structure score,* score*, the problem of finding the maximum scoring network, that is,*

$$\mathcal{G}^* = \arg\max_{\mathcal{G} \in \mathcal{G}_d} \text{score}(\mathcal{G} : \mathcal{D}),$$

is NP-Hard for any $d \geqslant 2$, where $\mathcal{G}_d = \{\mathcal{G} : \forall i, |\text{Pa}_{X_i}^{\mathcal{G}}| \leqslant d\}$.

In other words, finding the maximal scoring network structure with at most d parents for each variable is NP-hard for any d greater than or equal to 2 [Chickering *et al.* 1994; Chickering 1996; Chickering *et al.* 2004; Suzuki 2017]. This is because of the super-exponantial search space that one has to traverse to obtain the maximal scoring network. Koivisto and Sood [2004]; Singh and Moore [2005]; Silander and Myllymaki [2012] provide a detailed description and proof of NP-Hardness of this combinatorial problem.

There has been much work to further explore the repercussions of the result in Theorem 3.4. More specifically, learning a general graph structure is NP-hard for a bounded indegree less than or equal to d for the Bayesian score [Chickering 1996; Chickering *et al.* 2004]; polytrees (graphs with underlying tree structures) [Dasgupta 1999]; finding a path in a graphical model [Meek 2001]; and even when certain properties of the problem are known (eg. perfect independence, generative, inference, or information oracle) [Chickering *et al.* 2004].

The result in Theorem 3.4 might be discouraging. However, due to the developments of local search procedures we are able to provide a solution using a heuristic hill-climbing search.

Chickering *et al.* [1995]; Buntine [1991] proposed a local search over graph structures. The intuition of the heuristic local search procedure is as follows. Suppose we have a arbitrary candidate network as depicted at the center of Figure 25 labeled (a). We can decide to perform local perturbations in an attempt to improve the network structure relative to the data and a particular score. Suppose that we are given the following options: to reverse the edge $X_2 \to X_5$ obtaining network (b) which gives us a score of 75; to delete the edge $X_6 \to X_5$ obtaining network (c) which gives us a score of 90; to add an edge $X_4 \to X_2$ obtaining network (d) which gives us a score of 50; or to reverse the edge $X_5 \to X_3$ obtaining network (e) which results in a cyclic network.

Obviously we do not consider network (e) since this is not a legal Bayesian network. The most favorable transition would be to delete the edge $X_6 \rightarrow X_5$. This option improves the current network score from 70 to 90. It is not clear whether this transition will be favorable in the long term, however, it does provide a better network structure from a single transition.

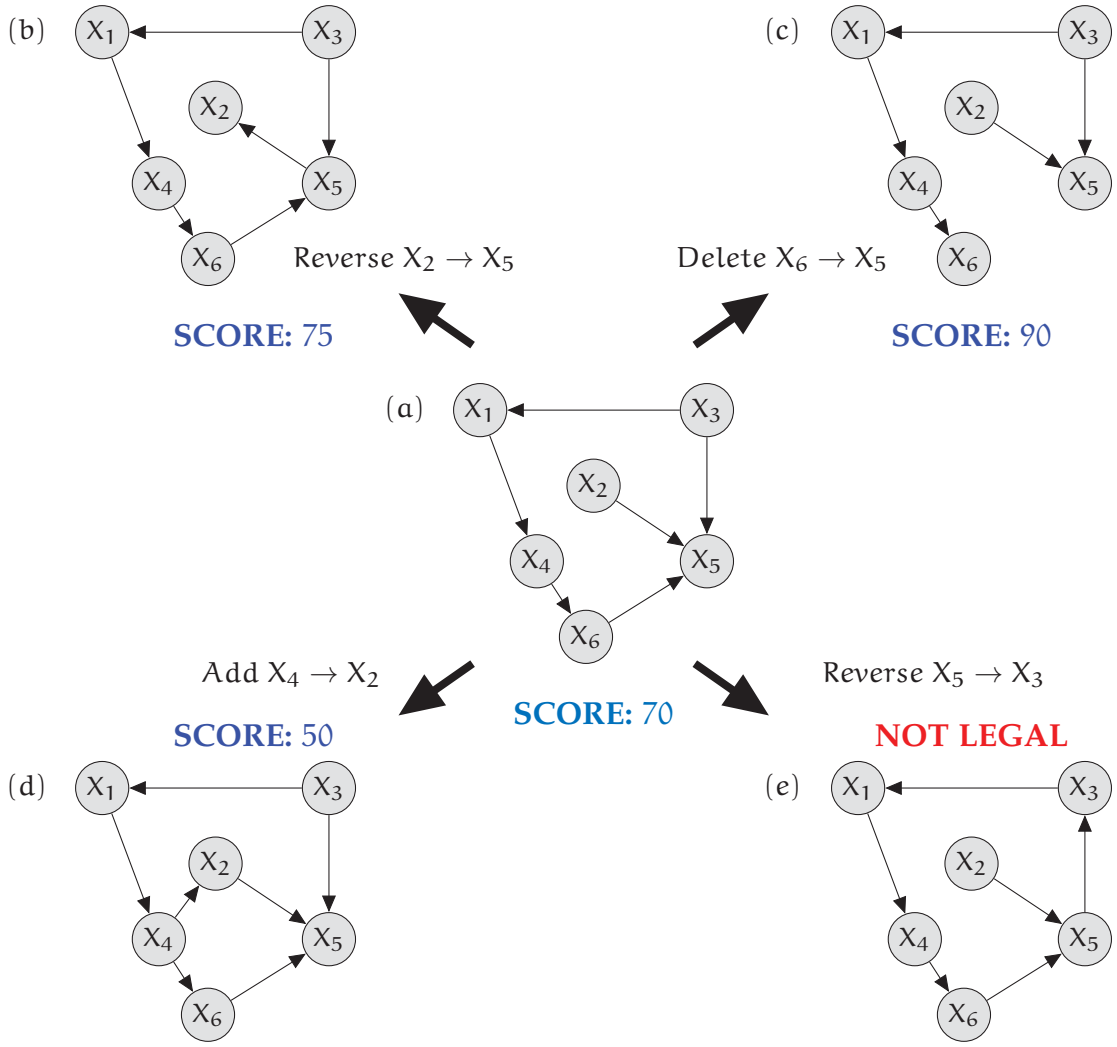

Figure 25: An illustration of the huristic search procedure between different candidate Bayesian network models. This figure shows local perturbations in an attempt to improve the network structure relative to the data and a particular score. Network (a) provides the current network structure. Network (b) reverses the edge $X_2 \rightarrow X_5$ which increases the network score by 5. Network (c) deletes the edge $X_6 \rightarrow X_5$ which increases the score by 20. Network (d) adds an edge $X_4 \rightarrow X_2$ which decreases the score by 20. Finally, network (e) reverses the edge $X_5 \rightarrow X_3$ obtaining an illegal network structure.

There are two main design choices that one needs to make when performing a local structure search: the choice of search operators and search procedure. Firstly, we must select a set of operators which are local steps to traverse the search space. Common choices for local search operators are edge addition, reversal and deletion.

The edge reversal search operator might seem counter intuitive since it can be achieved by a simple edge deletion and edge addition. When optimising that struc-

ture score over the search space, deleting an edge with the intention to perform a edge reversal would lower the overall structure score for the next step. Therefore, having an edge reversal transition allows us to explore the option of reversing edges without encountering this issue.

There are several other search spaces one could consider such as moving a variable to a new position in the network [Moore and Wong 2003]; searching over ordering and bounded in-degrees [Teyssier and Koller 2012]; or perturbing the data out of local optimum [Elidan *et al.* 2002]. Although methods which take larger steps in the search space may be expensive, they are empirically faster and more resistant to local optimum [Chickering 1996 2002; Elidan *et al.* 2002; Teyssier and Koller 2012].

The second design choice is to select a search technique that traverses the search space. Some choices include greedy hill-climbing, best first search, or simulated annealing. The most common choice is greedy hill-climbing (GESS) which starts with a prior network. The prior network could be an empty network; a best tree obtained from the procedure mentioned in Section 3.5; a random network; or one elicited by an expert. From this prior network we iteratively try to improve the network by utilising search operators. In greedy hill-climbing we always apply the change that improves the score until no improvement can be made. A comparison was done on various local search procedures by Chickering *et al.* [1995], such procedures included the K2 algorithm, local search, and simulated annealing. Chickering *et al.* [1995] note that local search provides the best time-accuracy trade-off to all other methods compared. Cooper and Herskovits [1992] were perhaps the first authors to propose a general graph search called the K2 algorithm. The K2 algorithm used an ordering over variables which would permit families of variables to specify an orientation.

There are two major issues with the greedy local search procedure. The resulting network structure can be interpreted as a local optimum for which no operator can improve the network score. Suppose we started the local search procedure at position C in Figure 26. We might unfortunately arrive at a poor local maximum, position A, relative to the global maximum, position B, which is considerably better.

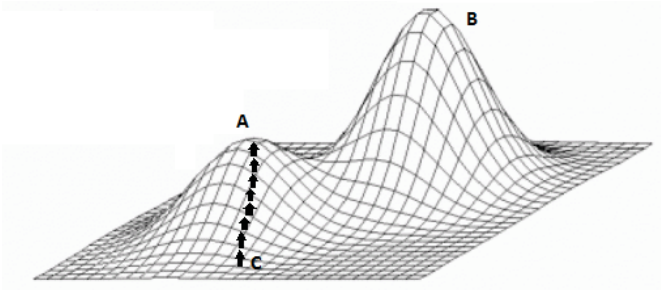

Figure 26: An illustration of a local search procedure which takes small steps in the search space and arrives at a local optimum [Tompkins and Lawley 2009].

The second problem arises when we encounter a plateau in the search space. Consider Figure 27, at position B where if we look around the current structure there may be a variety of possible network transitions which give the same score. In this case there is no way to know which direction to proceed towards.

This occurs frequently in Bayesian network structure learning because of I-equivalent network structures yielding the same score given a score-equivalent structure score

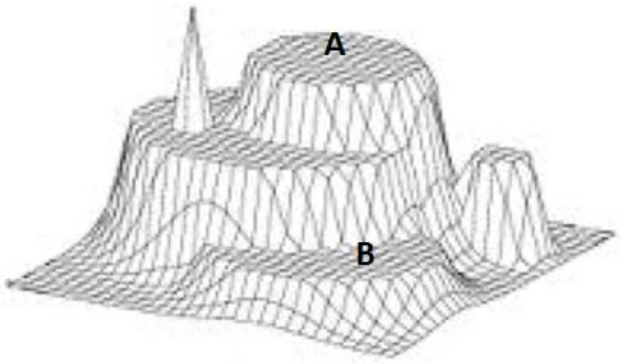

Figure 27: An illustration of a local search procedure which takes small steps in the search space and arrives at a local optimum [Tompkins and Lawley 2013].

(eg. those in Figure 8). In order to avoid these issues in practice we augment greedy hill-climbing with two search techniques: random restarts and tabu lists.

RANDOM RESTARTS: In random restarts, when we reach a local optimum we take k random steps and then continue traversing the search space using the search procedure. The intuition is that if we are at a local maximum that is shallow then k random steps will set us up in a better position to explore a better optimum.

TABU LISTS: In tabu lists we try to avoid treading the same path over and over again by maintaining a list of the most recent k steps taken by the search procedure. Glover and Laguna [2013] provide a detailed discussion on the effects of tabu lists on structure search procedures.

Figure 28 presents the effects of learning parameters only and learning both parameters and structure on the ICU alarm network presented in Figure 22 [Koller and Friedman 2009]. The x-axis shows the number of samples (M) and the y-axis shows the distance measured as K-L divergence [Minka 2005] between the learned and true distribution. The solid line in the figure indicate both structure learning and parameter estimation and the dotted line indicates learning only parameters given the true structure.

The result in Figure 28 suggests that the structure learning problem is not intrinsically more difficult than the parameter estimation problem given observable data. However, learning from synthetic data has a much stronger signal for the correlation of structure between variables than in real data.

Other examples of related work using local search procedures and traversal algorithms including a search over I-equivalent classes [Chickering 1996 2002]; and constraint-based algorithms which guarantee obtaining the correct network at a large sample limit, such as the SGS [Spirtes *et al.* 2000] and KES [Nielsen *et al.* 2002] algorithms Höffgen [1993] attempted to assess the rate of learning over the number of samples for Bayesian network structures with bounded indegree. Höffgen [1993] presents that relative entropy of the ground truth, if we learning with the number of variables, grows logarithmically with the number of samples. Similar work also

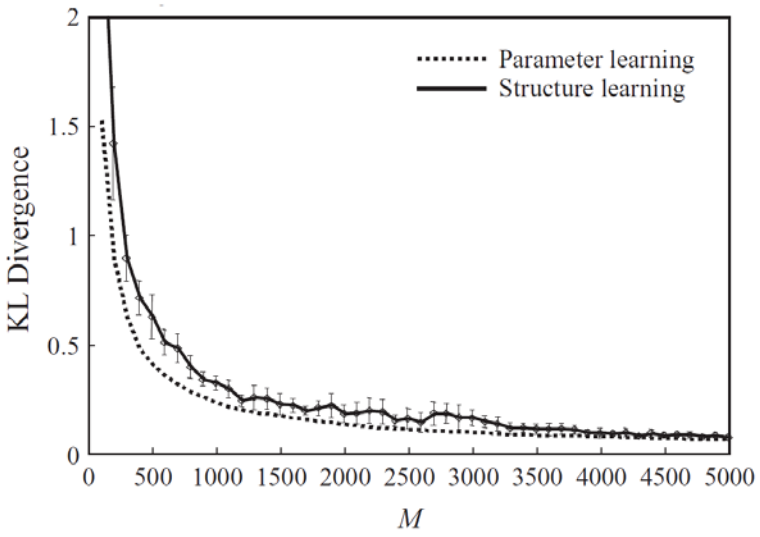

Figure 28: The performance of two learning tasks for Bayesian networks. Adapted from Koller and Friedman [2009]. The dotted line represents parameter estimation given the true structure, and the solid line represents structure and parameter learning.

explores the effects of using structure scores with penalties [Friedman and Yakhini 1996]. Abbeel *et al.* [2006] present a polynomial-time algorithm to learn a low-degree maximum-likelihood Bayesian network. Learning the structure for dynamic Bayesian networks [Friedman *et al.* 1998] and object-relational models [Friedman *et al.* 1999; Getoor *et al.* 2002] have also been proposed.

In the next section we analyse the algorithmic complexity of this learning problem and present some literature which attempts to reduce the computational burden of local search procedures by a clever use of data structures.

3.7 STRUCTURE LEARNING COMPLEXITY

Learning the structure of a Bayesian network can be viewed as a combinatorial optimisation problem where we attempt to select a structure through a search procedure using a scoring function over a search space. In this section we explore several techniques to reduce the total structure search complexity by taking advantage of score decomposability.

Recall Figure 25 where we evaluate several transitions from an initial network structure using various search operators. The computational cost of this structure search procedure is as follows. Suppose we have a Bayesian network with n nodes, then there exists $n(n-1)$ possible edges. Each edge is either present in the network or absent. Edges which are present can be deleted or modified using edge removal or reversal, and edges which are absent can be added using an edge addition. Therefore, there are asymptotically $O(n^2)$ total operators which we consider at each search step.

To evaluate the score of a network with respect to a selected scoring function we need to calculate the score of n families in the network. For each family we need to calculate the sufficient statistics for the scoring function which takes a traversal

through the training data, $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$. This traversal will take $O(M)$ steps. Therefore, it would take $O(nM)$ to evaluate the score of a candidate network.

In order to avoid illegal networks during searching, such as network (e) in Figure 25, we need to perform an acyclicity check on every potential transition network structure. This can be achieved with a simple breath-first search which requires $O(E)$, where E is the number of edges in the network.

Therefore in total, we require $O(Kn^2(Mn + E))$ to perform the structure search algorithm, where K is the total number of steps to the local optimum structure. For even moderately sized networks (eg. 30 to 60 variables) the computation becomes unmanageable.

The above presents a significant problem in terms of the general complexity of the structure search algorithm. However, we can exploit the decomposability property of the score function for significant computational savings.

Consider the transition from network (a) to (d) in Figure 25. Recall Proposition 3.1 which illustrates the decomposability of the likelihood score. The score of network (a) can be expressed using the score decomposability property as

$$\begin{aligned}\text{Score}(X_1, X_2, X_3, X_4, X_5, X_6) = &\text{Score}(X_1|X_3)\\ &+ \text{Score}(X_2)\\ &+ \text{Score}(X_3)\\ &+ \text{Score}(X_4|X_1)\\ &+ \text{Score}(X_5|X_2, X_3, X_6)\\ &+ \text{Score}(X_6|X_4).\end{aligned}$$

The score for network (d) in Figure 25, decomposes as

$$\begin{aligned}\text{Score}(X_1, X_2, X_3, X_4, X_5, X_6) = &\text{Score}(X_1|X_3)\\ &+ \text{Score}(X_2|X_4)\\ &+ \text{Score}(X_3)\\ &+ \text{Score}(X_4|X_1)\\ &+ \text{Score}(X_5|X_2, X_3, X_6)\\ &+ \text{Score}(X_6|X_4).\end{aligned}$$

The scores of networks (a) and (d) are exactly the same except for variables X_2's family. In network (a) we have $\text{Score}(X_2)$ and in network (d) we have $\text{Score}(X_2|X_4)$. Thus we need only recalculate the family score for X_2. Thus, each transition in the search space can be further expressed in terms of a $\triangle$score between each respective network given the search operators. This notion of a $\triangle$score can also be extended for edge reversal and deletion.

Exploiting the decomposability property allows us to compute one or two family scores at each transition, as opposed to recalculating every family score. Recall that to calculate a family score we need to calculate sufficient statistics for each variable in the family. This would take $O(nM)$ as opposed to $O(n^3M)$. Other methods that attempt to reduce the computational burden in structure learning literature include

caching sufficient statistics and the use of data structures such as priority queues [Moore and Lee 1998; Deng and Moore 1995; Moore 2000; Komarek and Moore 2000; Indyk 2004].

In this chapter we reviewed various structure scores; a method to learn a tree-structured network; and local search techniques to recover graph structures. We also explored methods for computational saving given the intractability of the search procedure to traverse the search space. In the next chapter we use the contributions in this and the previous chapter to introduce influence between stochastic processes. We then begin to develop procedures to recover influence relations between them.