MAC6916 PROBABILISTIC GRAPHICAL MODELS
LECTURE 6: VARIABLE ELIMINATION

DENIS D. MAUÁ

1. Introduction

We resume our study on algorithms for computing the probability of evidence (or the partition function) of a Bayesian network or Markov network. Here, we will consider a slightly more general version of the problem, stated as follows. Given potentials $\phi_1, \ldots, \phi_m$ over a set of variables $\mathcal{V}$ and a set $\mathcal{X} \subseteq \mathcal{V}$, compute

$$\sum_{\mathcal{V}-\mathcal{X}} \prod_j \phi_j \,. \qquad \text{[sum-product problem]}$$

For even moderately sized sets $\mathcal{V}$, the above expression is impractical to compute by applying the operations as they appear. We have seen that this problem can be solved by sampling algorithms, but they do not provide any guarantees with finite time. In this lecture, we will analyze one of the most popular exact algorithm for that problem, called variable elimination.

Variable elimination is a simple and intuitive algorithm for manipulating information that seems to have been re-developed many times in different areas such as operational research, constraint satisfaction, relational databases, and probabilistic inference, under many different names (and subtle variations). For instance, it was used for automating pedigree analysis in genetics [3], and for computing conjunctive queries in relational databases [1]; it was discussed as a sort of non-sequential dynamic programming paradigm in the operational research field [2]; it underlies the basic version of the Davis-Puttnam algorithm for solving satisfiability [4]; it was one of the first practical exact algorithms for computing marginals (i.e., belief updating) in Bayesian networks [5, 6]. Dechter [7] was one of the first persons to note the similarity of all these methods, and to formalize them as a unifying algorithm (with a few minor modifications and under the name of bucket elimination). The algorithm's key insight is to exploit the distributivity of addition (marginalization) over multiplication (product) to decompose the problem into smaller problems in a dynamic programming fashion.

2. An Algebra of Potentials

Variable elimination is a generic framework which can be used to solve many computational problems that consist in aggregating pieces of information or knowledge and projecting into a new domain (this include problems that are conceptually unrelated to probabilistic inference such as consulting a relational database). Before presenting variable elimination over the concrete objects that we manipulate

Date: March 29, 2016.

(real-valued potentials), we first analyze the algebraic properties of potentials at an abstract level. To this end, let $\mathcal{V}$ be a finite set of variables (with their respective domains).

labeled abstract potential
scope

Definition 1. A *labeled abstract potential* is an object ϕ with an associated *scope* $\mathcal{S} \subseteq \mathcal{V}$. We denote by $\mathrm{sc}(\phi) = \mathcal{S}$ the function that maps labeled abstract potentials to their scopes, and often write a labeled abstract potential as $(\phi, \mathcal{S})$.

For convenience, we will refer to labeled abstract potentials simply as potentials (bearing in mind that these need not to refer to the concrete potentials we have defined previously). Let Φ be the set of all abstract potentials.

combination

Definition 2. A *combination* is a binary operation $\times$ on Φ satisfying $\mathrm{sc}(\phi \times \psi) = \mathrm{sc}(\phi) \cup \mathrm{sc}(\psi)$.

In our concrete potentials, combination corresponds to product of functions.

elimination

Definition 3. A *elimination* is a mapping from $\Phi \times \mathcal{V}$ to Φ such that $\mathrm{sc}(\phi - X) = \mathrm{sc}(\phi) - \{X\}$.

Elimination of concrete potentials is usually defined as the variable elimination of the first operand from the second one.

An algebra of (labeled abstract) potentials is a system $(\mathcal{V}, \Phi, \times, -)$ closed under combination and elimination, and satisfying the following axioms [12, 11]:

$$\phi \times \psi = \psi \times \phi \qquad \text{[commutativity of combination]}$$
$$\phi_1 \times (\phi_2 \times \phi_3) = (\phi_1 \times \phi_2) \times \phi_3 \qquad \text{[associativity of combination]}$$
$$\phi - X - Y = \phi - Y - X \qquad \text{[transitivity of elimination]}$$
$$X \notin \mathrm{sc}(\phi) \Rightarrow (\phi \times \psi) - X = \phi \times (\psi - X) \qquad \text{[distributivity of } - \text{ over } \times\text{]}$$

Let $\phi_1, \ldots, \phi_n$ be potentials and X be a variable such that $X \notin \mathrm{sc}(\phi_1) \cup \cdots \cup \mathrm{sc}(\phi_m)$, for some $m < n$. These axioms allows us to derive:

$$(\phi_1 \times \cdots \times \phi_n) - X = \phi_1 \times \cdots \times \phi_m \times (\phi_{m+1} \times \cdots \times \phi_n - X)$$

Repeated application of the result above suffices to derive a sound algorithm for the elimination of a set of variables from a combination of potentials. Our interest in the algebra of potentials is to compute inference in discrete probabilistic graphical models (Bayesian networks and Markov networks). Nevertheless, many other representation devices can be shown to satisfy these axioms, and hence, variable elimination is well-defined on them. Examples include Gaussian distributions, systems of linear equations, convex sets, Boolean algebra, and belief functions [11]. Notably, product and maximization of discrete functions (i.e., potentials) satisfy the axioms; we can thus use variable elimination to find maximum probability configurations.

3. Variable Elimination

valuation

The typical implementation of variable elimination operates over potentials and sum-marginalization. Recall that a *valuation* is a function mapping each variable to a value in its, and a partial valuation is a valuation that is defined on a proper subset of the variables.

concrete potential

Definition 4. A *(concrete) potential* is a non-negative mapping on the valuations annotated with a set of variables $\mathcal{S}$ called its scope. The mapping satisfies $\phi(\nu) = \phi(\nu')$ for any two valuations ν and ν' that agree on $\mathcal{S}$ (i.e., $\nu(X) = \nu'(X)$ for all $X \in \mathcal{S}$).

Intuitively, the scope is the (not necessarily minimal) set of variables on which the potential depends. We denote the value returned by ϕ on a valuation ν by $\phi(\nu)$, and the scope of ϕ by $\mathrm{sc}(\phi)$. We sometimes write $\phi(\mathcal{S})$ to represent the fact that $\mathcal{S}$ is the scope of $\mathrm{sc}(\phi)$.

Definition 5. The combination of potentials ϕ and ψ is the potential $\phi\psi$ with scope $\mathrm{sc}(\phi) \cup \mathrm{sc}(\psi)$ such that $(\phi\psi)(\nu) = \phi(\nu)\phi(\nu)$ for any valuation ν on $\mathrm{sc}(\phi\psi)$.

elimination

Definition 6. The *elimination* of a variable X from a potential ϕ is the potential $\phi - X$ with scope $\mathrm{sc}(\phi) - \{X\}$ such that $(\phi - X)(\nu) = +\{\phi(\nu') : \nu'(Y) = \nu(Y)$ for all $Y \in \mathcal{V} - \{X\}\}$.

The Variable Elimination Algorithm is a procedure for computing marginals of a combination of potentials. The algorithm takes a set

$$\Psi = \{\phi_1, \ldots, \phi_m\}$$

of potentials, and an *ordered* set of variables $(X_1, \ldots, X_n)$ to be eliminated, and returns

$$(\phi_1 \times \cdots \times \phi_m) - X_1 - \cdots - X_n \,.$$

It proceeds as follows

(1) For $i \leftarrow 1$ to $i \leftarrow n$ do:
   (a) Collect all potentials $\phi \in \Psi$ such that $X_i \in \mathrm{sc}(\phi)$ and store them in Φ_i
   (b) Combine all potential $\phi \in \Phi_i$ and then eliminate X to obtain ψ_i
   (c) Remove the potential $\phi \in \Phi_i$ from Ψ and add potential ψ_i
(2) Return $\times\{\phi \in \Psi\}$

Let us analyze the time complexity of the algorithm. A naive implementation of Step (a) requires considering each potential in Ψ, and performing a linear search over its scope. Let ω be the maximum cardinality of a scope in Ψ (at any time). Then assuming that insertion in Φ_i takes constant time, this step takes $O(m \cdot \omega)$ time (since the size of Ψ never increases at each iteration). Consider now Step (b). Combining two potentials ϕ and ψ can be done by considering all partial valuations defined over the variables in $\mathrm{sc}(\phi\psi)$ and for each valuation performing the product $\phi(\nu)\psi(\nu)$. Let c be the maximum cardinality of a variable; assuming multiplication is performed in constant time, combination takes $O(c^k)$ time where $k = |\mathrm{sc}(\phi) \cup \mathrm{sc}(\psi)|$. Elimination can be performed similarly with the same asymptotic cost. Hence, Step (b) takes time $O(m_i \cdot c^{\omega_i+1})$, where $m_i = |\Phi_i|$ and $\omega_i = |\mathrm{sc}(\psi_i)|$. Assuming removal from Φ_i takes constant time, Step (c) can be performed in time $O(m_i)$. We also need to account for the combination of potentials in the last step and the size of the output. For simplicity, we will assume that this step takes constant time (as it is often the case in practice), which implies that all but a few variables are eliminated. The overall running time is thus

$$O\left(m\omega + \sum_{i=1}^{n} m_i c^{\omega_i}\right) = O\big((m+n)c^{\omega+1}\big)\,,$$

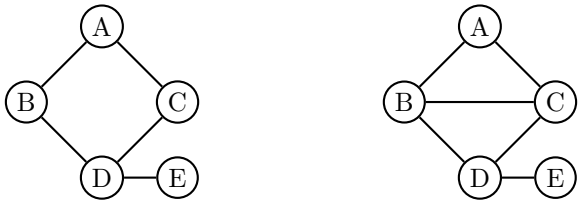

FIGURE 1. Left: Domain graph of the potentials $\phi(A, B)$, $\phi(A, C)$, $\phi(B, D)$, $\phi(C, D)$, $\phi(D, E)$. Right: The elimination graph according to elimination ordering A, B, C, D, E.

where we used the fact that sum of all m_i is at most m (since at each iteration we remove m_i potentials and add one, so that $m_n \leq m - \sum_{i=1}^{n-1}[m_i + 1]$.), and $\omega = \max_i \omega_i$.

elimination ordering

Note that while m, n and c are parameters of the model, the value ω is not. Moreover, ω depends on the *variable elimination ordering*. To see this, consider binary variables A, B, C and potentials $\phi_1(A, B)$ and $\phi_2(B, C)$. If we perform variable elimination using the ordering A, B we end up with

$$\psi_1(B) = \phi_1(A, B) - A\,, \quad \psi_2(C) = (\phi_2(B, C)\psi_1(C)) - B\,,$$

and obtain that $\omega = 1$. On the other hand, if we use the ordering B, A, the algorithm generates

$$\psi'_1(A, C) = \phi_1(A, B)\phi_2(B, C) - B\,, \quad \psi'_2(C) = \psi_1(A, C) - A\,,$$

and thus $\omega = 2$. Can we obtain a bound on ω as a function of the model (the potentials and variables) and the elimination ordering? Not surprisingly, we will represent the model graphically and determine ω as a function of the graph.

domain graph

Definition 7. The *domain graph* of a set of potentials Ψ is an undirected graph whose nodes are the variables appearing in the scope of a potential in Ψ and an edge connects two nodes if and only if the corresponding variables appear in the same scope.

Figure 1(left) shows an example of a domain graph for a set of pairwise potentials. It should be clear that the domain graph of a Markov network is its own graph, and that the domain of a Bayesian network is its moral graph.

For the rest of this discussion, we will assume that the elimination ordering is complete, that is, that it contains all the variables appearing in a potential (the following results are still valid if we elimination all but one variable, and can be adapted to the general case to account for the complexity of the output). Note that when the elimination ordering is complete, the output is a number (instead of a potential).

Definition 8. Let $D = D_0, D_1, \ldots, D_n$ be the domain graphs of the sets Ψ obtained during the execution of variable elimination (D is the domain graph of the initial set). The *elimination graph* is an undirected graph over the nodes of D such that there is an edge if and only if that edge appears in at least one of $D, D_1, \ldots, D_n$.

elimination graph

The elimination graph depends on the elimination ordering used. For example, consider again the potentials $\phi_1(A, B)$ and $\phi_2(B, C)$. Their domain graph is the

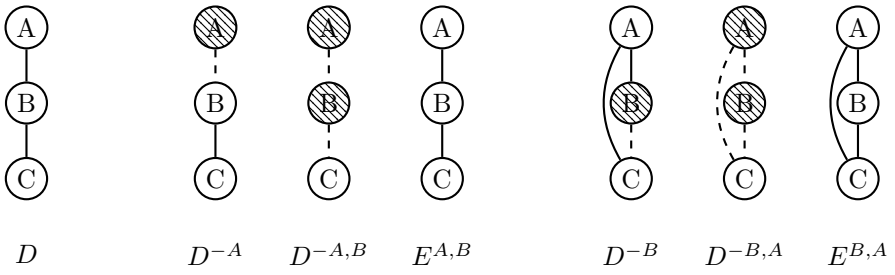

FIGURE 2. Domain graph of eliminations of the potentials $\phi(A, B)$ and $\phi(A, C)$ for the elimination orderings A, B (left) and B, A (right).

leftmost graph D in Figure 2. After eliminating variable A, we are left with potentials $\psi_1(B)$ and $\phi_2(B, C)$, whose domain graph is D^{-A} (ignoring shaded nodes and dashed edges). Eliminating B then generates the graph $D^{-A,B}$. The elimination graph of this run is the graph $E^{A,B}$, which equals the initial domain graph.

Now consider a new run of the algorithm where we first eliminate B, obtaining $\psi_1'(A, C)$. The respective domain graph is D^{-B}. Note that an edge between A and C was inserted, representing the potential ψ_1'. Eliminating variable A generates the graph $D^{-B,A}$ in the figure. The rightmost graph $E^{-B,A}$ is the elimination graph of this ordering; we can see that this graph is more connected than the initial domain graph.

It is not a coincidence that in our example the value of ω is the size of the largest maximal clique in the elimination graph of each run minus one.

Definition 9. Consider a graph G and a node X. The *elimination* of node X from G is the graph G^{-X} obtained by pairwise connecting the neighbors of X and removing X and its incident edges. node elimination

For example, the graph D^{-A} in Figure 2 is obtained by eliminating variable B from the (initial) domain graph D. As the following result shows, elimination of variables can be represented graphically by elimination of nodes in the corresponding domain graph.

Lemma 1. *The graph D_i is the graph obtained by eliminating variable X_i from D_{i-1}.*

Proof. Let Ψ_i denote the set Ψ after the ith iteration. Then, $\Psi_i = \Psi_{i-1} - \Phi_i \cup \{\psi_i\}$. By construction, X_i is not in D_i. First note that the neighbors of X_i in D_{i-1} are $\bigcup_{\phi \in \Phi_i} \text{sc}(\phi)$, and this set is the scope of ψ_i. Since ψ_i is Ψ_i, the neighbors of X_i must be pairwise connected in D_i. It remains to show that no other edge is inserted or removed. Let Y and Z be adjacent variables in D_{i-1} which are not neighbors of X_i. By definition of domain graph, there must be a potential in Ψ_{i-1} whose scope contains both Y and Z and does not contain X. By construction, that potential is also in Ψ_i (because it is not in Φ_i), hence Y and Z are adjacent in D_i. Thus, every edge in D_i which is not incident in X_i or in one of its neighbors is also in D_{i+1}. Now consider an edge Y, Z in D_i. Then either Y and Z appear in ψ_i are neighbors of X_i or else they appear in some potential which is also in Φ_i. □

That result extends to the elimination graph (and justifies its name).

Corollary 1. *The elimination graph equals the graph obtained as follows: Start with D and for $i = 1, \ldots, n$ connect any two neighbors X_j and X_k of X_i such that $i < j < k$.*

Thus, the elimination graph is obtained by *simulating* the elimination of each node sequentially, but with marking instead of removing of the eliminated nodes.

Proof. It follows from Lemma 1, by noting that the neighbors of X_i in D_{i-1} are the neighbors of X_i in D that have not been eliminated, hence are some variable X_j with $j < i$. □

We can now state the connection between the graphical representation of the model and the complexity of variable elimination.

Lemma 2. *The value ω_i is the size of the neighborhood of X_i in D_{i-1}, which is also the number of higher-ordered neighbors in the elimination graph.*

Proof. To see that ω_i is the number of neighbors of X_i in D_{i-1}, note that the scope of ψ_i contains exactly the variables which co-appear with X_i in the scope of a potential (hence are neighbors of X_i in D_{i-1}). □

We parametrize the class of elimination graphs by the size of the largest clique.

induced width

Definition 10. The *induced width* of an elimination ordering is the the size of the largest clique of the corresponding elimination graph minus one.

We can now relate the size of generated potentials with the elimination graph.

Proposition 1. *The induced width of an elimination order equals $\omega = \max_i |\psi_i)|$.*

Proof. First note that the scopes of the initial potentials are cliques in the initial domain graphs and hence cliques in the elimination graph. The $\mathrm{sc}(\psi_i) \cup \{X_i\}$ is also a clique in the elimination graph, since $\mathrm{ne}(X_i)$ is a clique in D_i, X_i is connected to them in D_{i-1}. To see that these cliques are maximal, note that an edge occurs only between variables in the scope of a common potential, and that the scope of any other potential generated is a subset of $\mathrm{sc}(\psi_i) \cup \{X_i\}$. □

We are now ready to state the complexity of variable elimination with respect to the graphical representation of its input. Let b be the smallest cardinality of a variable.

Theorem 1. *Variable elimination takes time $\Omega(mb^{\chi})$ and $O((m+n)c^{\omega+1})$, where χ is the size of the largest clique in the domain graph, and ω is the induced-width of the elimination ordering plus one.*

4. Finding Good Elimination Orderings

According to Theorem 1, the complexity of the algorithm is lower bounded by the biggest clique size in the domain graph and upper bounded by the biggest clique size in the elimination graph. The former is part of the input, and cannot be improved. As we have seen by example, the latter depends on the elimination ordering. Thus, we want to find the elimination ordering that minimizes the induced width.

treewidth

Definition 11. The *treewidth* of a graph G is the minimum induced width of an elimination ordering.

Thus, the optimum implementation of variable elimination is to select an elimination ordering whose induced width equals the treewidth. Unfortunately, finding such an ordering is an NP-hard problem (verifying if there is an ordering whose induced width is at most a certain integer is NP-complete) [13]. Instead, we will devise efficient heuristics for finding good elimination orderings (elimination orderings with relatively small induced width).

There is an interesting connection between elimination orderings and chordalization of graphs.

Definition 12. A chordal graph H is a *chordalization* of a graph G if every edge in G is also in H. chordalization

The following result shows that an elimination can be seen as a chordalization of the (initial) domain graph.[1]

Proposition 2. *The elimination graph is chordal.*

Proof. We prove by contradiction. Take a chordless cycle $X_{i_1}, \ldots, X_{i_k}$, ordered according to the elimination ordering (this is always possible, since a cycle is closed under shifting). Eliminating X_{i_1} generates an edge between X_{i_2} and X_{i_k} in the elimination graph, and hence a chord in the cycle. □

Hence, variable elimination can be seen as a means of inserting edges in the domain graph so as to make it chordal. For instance, the chordal graph in Figure 1(right) is the elimination graph of the non-chordal domain graph on the left, obtained by eliminating nodes in the ordering A, B, C, D, E. The edges in the elimination graph which are not in the domain graph are called *fill-in edges*. Each fill-in edge denotes the increase of the scope of a potential with respect to the scopes in the input potentials, which implies an exponential increase in the time complexity of the algorithm. The edge $B - C$ in the graph in Figure 1(right) is the single fill-in edge. fill-in edges

It can be shown that every chordalization corresponds to an elimination. Hence, the treewidth of a graph can be equivalently stated as the minimal chordalization of a graph (where minimality is taken with respect to the number of edges). Consequently, the complexity of variable elimination is minimized by finding the smallest chordalization of the domain graph (where the size of a chordalization is the number of fill-in edges). This suggest the *min-fill heuristic*, which starts with the domain graph and then recursively selects the node whose elimination leads to the minimum number of fill-in edges and eliminates that node. Let $\text{fill}(G - X)$ denote the number of fill-in edges introduced by eliminating X from G (i.e., the number of edges needed to turn $\text{ne}(X)$ into a clique). The min-fill heuristic is: min-fill heuristic

(1) Start with the domain graph D and an empty list L
(2) For $i \leftarrow 0$ to $i \leftarrow n$ do:
    (a) Select the node X_j in D that minimizes $\text{fill}(D - X)$
    (b) Add X_j to the end of L and update $D \leftarrow D - X_j$
(3) Return L

The algorithm takes time linear in the number of nodes and quadratic in the largest clique size if the graph is represented as an adjacency list (due to computing the number of fill-in edges, which requires checking whether neighbors are connected).

[1] Chordalization is also called triangulation.

While the min-fill heuristic may seem intuitively a good approach, it does not always lead to the minimum width elimination ordering. It thus so when the domain graph is already chordal.

simplicial node

Definition 13. A node X is *simplicial* if $\text{ne}(X)$ is a clique.

Clearly, a node X is simplicial iff $\text{fill}(D - X) = 0$. Hence, min-fill will always prefer a simplicial node over some non simplicial nodel.

perfect elimination ordering

Definition 14. An elimination ordering is *perfect* if it introduces no fill-in edges.

Theorem 2. *The following statements are true.*

(1) Any chordal graph with at least two nodes has at least two simplicial nodes.
(2) A graph is chordal if and only if it admits a perfect elimination ordering.
(3) If D is chordal and X is simplicial then $D - X$ is chordal.
(4) A elimination ordering which repeatedly eliminate simplicial nodes is perfect.

According to the previous theorem, the min-fill heuristic produces a perfect elimination ordering in chordal domain graphs. The following result is useful to compute the induced width of an elimination ordering (recall that the elimination graph is chordal).

Proposition 3. *Let $D_0, D_1, \ldots, D_n$ be a sequence of graphs obtained by repeatedly eliminating simplicial nodes $X_1, \ldots, X_n$. Then the induced width of the ordering is the maximum number of neighbors of a variable X_i in D_{i-1}.*

The size of the potentials ψ_i created are related to the number of neighbors of the variable X_i being eliminated. Thus, it is reasonable to eliminate variables of small degree. The min-degree heuristic attempts to find a good elimination ordering by favoring variables with fewer neighbors. The algorithm is very similar to min-fill:

(1) Start with the domain graph D and an empty list L
(2) For $i \leftarrow 0$ to $i \leftarrow n$ do:
    (a) Select the node X_j in D that minimizes $| \neq X|$
    (b) Add X_j to the end of L and update $D \leftarrow D - X_j$
(3) Return L

Both the min-fill and the min-degree heuristic aim at minimizing the maximum size of the scope of ψ_i. However, the size of the scope also depends on the cardinality of the variables. For instance, it might be better to generate a potential with 10 binary variables than to generate a potential with 2 variables of cardinality 100. This leads us to consider weighted versions of these heuristics. The *weighted min-fill heuristic* selects a node that has the smallest sum of weights of fill-in edges, where the weight of an edge is the product of the cardinalities of the adjacent variables. The *weighted min-degree heuristic* selects a node that minimizes the weighted degree, defined as the product of the cardinality of the neighbors.

weighted min-fill heuristic

weighted min-degree heuristic

Note that the min-fill and the min-degree heuristics can be computed dynamically at each iteration from the corresponding graph D_i. This is usually a good approach when the elimination ordering heuristic is fixed. It might be interesting however to obtain an elimination order before calling variable elimination, by running several heuristics (possibly each multiple times) and select the best ordering.

5. Conclusion

Variable elimination is a simple algorithm for computing sum-product inferences in Bayesian networks and Markov networks. Its complexity is exponential in the induced width of the elimination ordering used. Finding an optimal elimination ordering is computationally difficult and greedy heuristics are used. These heuristics often perform well in practice, generating close to optimal orderings.

Variable elimination is particularly efficient in models of small treewidth. This is the case of tree-shaped Bayesian and Markov networks, and of polytree-shaped Bayesian networks of small in-degree. Tree-shaped models are routinely used to represent sequential data such as speech, text, and movies. For these graphs an optimal elimination ordering can be found efficiently.

6. Reading

There is no recommended reading this week. For more information variable elimination, check out chapter 6 of [8] and Chapter 9 of [9].

7. Exercises

No exercises this week.

8. Assignment

- Implement variable elimination and elimination ordering heuristics. Your algorithm should preferably take a Markov network in UAI format and a set of evidences and compute the probability of each evidence.
- Test the algorithm's performance on the provided set of Bayesian networks (generate evidence test cases).
- Write report describing implementation and discussing empirical analysis.

Deadline: April, 11.

References

[1] C. Beeri, R. Fagin, D. Maier and M. Yannakakis On the desirability of acyclic database schemes. *Journal of the Association for Computing Machinery*, volume 30, issue 3, pp. 479–513, 1983.

[2] U. Bertelé and F. Brioschi. Nonserial Dynamic Programming. Academic Press, 1972

[3] C. Cannings, E.A. Thompson and H.H. Skolnick The recursive derivation of likelihoods on complex pedigrees. *Advances in Applied Probability*, volume 8, issue 4, pp. 622–625, 1976.

[4] M. Davis and H. Putnam A Computing Procedure for Quantification Theory. *Journal of the ACM*, volume 7, issue 3, pp. 201–215, 1960.

[5] R. Shachter, B. D'Ambrosio, and B. Del Favero. Symbolic probabilistic inference in belief networks. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pp. 126–131, 1990.

[6] N. Zhang and D. Poole. A simple approach to Bayesian network computations. In *Proceedings of the 10th Biennial Canadian Artificial Intelligence Conference*, pp. 171–178, 1994.

[7] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, volume 113, issues 1–2, pp. 41–85, 1990.

[8] A. Darwiche. Modeling and reasoning with Bayesian networks. Cambridge University Press, 2009.

[9] D. Koller and N. Friedman. Probabilistic Graphical Models MIT Press, 2009.

[10] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, volume 60, issue 1, pp. 141–153, 1993.

[11] J. Kohlas. Information Algebras: Generic Structures for Inference. Springer Verlag, 2003.

[12] P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *Proceedings of the 4th Conference on Uncertainty in Artificial Intelligence*, pp. 169–198, 1988.
[13] S. Arnborg, D.G. Corneil and A. Proskurowski. Complexity of finding embeddings in a k-tree. SIAM Journal on Algebraic Discrete Methods, volume 8, issue 2, pp. 277–284, 1987.