Application of junction tree clustering methods for solving dynamic Bayesian networks probabilistic inference tasks

T V Azarnova and P V Polukhin
Department of Applied Mathematics and Informatics and Mechanics, Voronezh State University, Voronezh, Russia
E-mail: ivdas92@mail.ru

Abstract. The article covers clustering algorithms for dynamic Bayesian networks based on the construction of join tree. Investigated problems associated with simplifying the topology of dynamic Bayesian networks using clustering methods and consider various semantic approaches of junction trees construction. The paper presents a junction tree constructing algorithm for the dynamic Bayesian network that takes into account the transition and perception models applied to the process of network formation. Analyzed The relationship between the complete joint probability distribution for the dynamic Bayesian network and the complete joint distribution obtained for the junction tree. It is shown that the required complete joint distribution of the junction tree for the dynamic Bayesian network can be obtained as a product of local probability distributions for each node associated with the resulting junction tree, and proved that this distribution will be equivalent to the probability distribution of the original dynamic Bayesian network. Introduced the main structural approaches to the construction of junction tree for the discrete dynamic Bayesian network and inspected the use of junction tree for solving similar problems for continuous dynamic Bayesian networks with Gaussian and exponential distribution functions of network variables.

1. Introduction
Currently, tools based on dynamic stochastic models and particularly dynamic Bayesian networks (DBN) are widely used within the context of expert and intelligent systems for various object and scientific areas. Formalization of knowledge, setting parameters of dynamic Bayesian networks and directly obtaining probabilistic characteristics of the researched processes are based on the implementation of generic learning and probabilistic inference mechanisms. This approach makes it possible to reduce the sensitivity of models to a limited set of samples used in the training process, without reducing the total accuracy of obtained results. DBN in general is a set of static Bayesian networks (BN) taken in chronological order on a time interval \((t; t + k)\). In turn, the static BN is an acyclic directed graph, each node associated with a conditional probability table (CPT). The structure of the DBN can be formed on the basis of expert knowledge, when we have been known exactly which nodes used in the modeling process and conditional probabilistic relationships between these nodes are set [1]. Along with expert methods for constructing the structure of the DBN, we can use formalized procedures based on statistical methods of information processing, methods for testing statistical hypotheses about the presence of cause-effect relationships, and optimization methods for determining the direction of relationships. With a known structure, the main procedures for working with the DBN
are represented by network pooling and inference, which allow to determine the probability distribution for each of the network vertices based on the CPT and the evidences obtained for each time slice during the deployment of the DBN [2]. Clustering algorithms and stochastic algorithms based on the appendix of the Monte Carlo method can be used as network pooling algorithms. Among the most common algorithms for optimizing the BN structure and conducting network polling procedures are hypergraph and clustering algorithms. The clustering algorithms employment is the most adapted to the structure of the DBN, since it allows to provide sufficient consideration of transitive relationships between time slices. Clustering algorithms simplify the network structure just before a network pooling execution. In the framework of this research, we consider the issues of simplifying the DBN topology due to the clustering algorithms application and particularly the junction tree (JT) usage. The JT is a clustered graph obtained by converting a domain graph formed from the BN. The combination of the junction tree pooling procedure with Markov Chain Monte Carlo (MCMC) probability inference algorithms allows to make the pooling procedure more transparent and directed, to reduce the number of learn parameters by aggregation of network nodes, and also to exclude the possibility of obtaining ambiguous data during the algorithm execution. The apparatus based on the junction tree used for modeling both discrete and continuous DBN. The main goal of the research focused on the development of algorithms for constructing a junction tree for the DBN taking into account the features of its probabilistic semantics.

2. Building junction tree. Solving the problem of simplifying the structure of the dynamic Bayesian network. Implementing network pooling procedure

JT is a fairly common representation of BN for solving exact probabilistic inference problems. The transition from the initial structure of the BN to the JT allows to eliminate the polling stage of the original network, reduce the total number of stages required to calculate the probability distributions of each vertex in the BN. The advantage of the junction tree construction algorithm compared to the classical stochastic network pooling algorithms is their resource and time efficiency. Structurally, the JT can be represented as a set of multiple clicks \( C_i \), separated by the set of separators \( S_i \). The JT algorithm can be divided into two main stages: structure transformation and network polling. The static BN structure transforming procedure includes the following steps: domain graph, moral graph, triangular graph, union tree, and junction tree formation. The network pooling procedure allow to distribute messages between clicks \( C_i \) and \( n \) \( C_i \), separated by \( S_i \) and to determine the potentials \( \phi_i \) for each message during forward and backward pass.

Let’s consider explicitly main steps of building the structure of the junction tree for a static BN. The original BN denote as \( G \).

Procedure for building a domain graph. The domain graph \( G_m \) is a graph whose vertices and edges (arcs) coincide with the vertices and edges (arcs) of the original BN \( G \), but unlike the original BN, the domain graph does not contain conditional probabilities tables. The domain graph reflects only information about the presence or absence of cause-effect (parents, children) relationships between the BN variables without quantitative characteristics of existing dependencies. Thus, the domain graph is certain simplified representation of Bayesian network with coincide vertices of original graph \( G \) and without probabilities semantic [4].

The domain graph \( G_m \) moralization procedure. Moral graph is obtained from the domain graph by constructing additional arcs between any two vertices A and B, for which in the original BN there is a vertex C that is a child for both A and B. Moralization reflects the fact of an indirect relationship between the vertices A and B.

The moral graph triangulation procedure. A graph \( G_t \) is triangular only if it does not contain cycles of more than three vertices (we assume that a cycle is a set of vertices in which each vertex is connected exactly two other vertices of this set). The triangulation procedure can be performed as follows. A copy \( G_{nt} \) of the moral graph \( G_m \) is created. Until the vertex set of the graph \( G_{nt} \) is not empty, select the
vertex $V$, having the smallest number of neighboring vertices (if there are several vertices, selects any of them, or with minimum weight, where weight refers to the product of the number of possible values that can take observed variable and the number of possible values that can take its all neighboring vertices). All vertices that are neighbors toward to $V$ are connected, each neighbor of $V$ must be connected to all other neighbors of $V$. The same associations are also added to the graph $G_m$. Then remove $V$ from $G_m$ and return again to the vertex $V$ selection step. When the graph $G_m$ becomes empty, the graph $G_n$ becomes triangular, denote it as $G_t$.

The junction tree construction procedure. To build a junction tree, we must first create a union tree. The union tree has cliques and separators as vertices and differs from the junction tree by absence of the probabilistics description. During union tree $T_e$ construction process, we enter some counter $i = 1$, select node $X$ from the triangular graph $G_{i}$, which has the minimum number of neighboring vertices. If there are several vertices satisfying this condition, we can choose the vertex with the minimum weight. The set of vertices involves selected vertex, all neighboring to it and each other vertices are called as clique $C_i$ of triangular graph $G_{i}$. The set of vertices neighboring to vertex $X$, but have shared neighbors not belonging to the set $C_i$, form separators $S$. After the procedure of separators formation, remove all neighboring to $X$ vertices that do not belong to the set of separators $S$, remove the top vertex $X$, increment counter $i = i + 1$ and move to the next vertex. Repeat this procedure until the triangular graph is not empty. After the triangular graph $G_t$ becomes empty, the formation of the union tree structure begins directly. We connect each node of the clique $C_i$ with the separator $S$ and then $S$ with $C_j$, but under the assumption that $S$ is a subset of $S_i \subseteq C_j$. Consequently, we have union tree, containing nodes produces by connecting cliques and separators of graph $G_t$. The expression describing the total weight of the union tree is determined by the sum of all separators $S_i$ and has the following form

$$w(T_e) = \sum_{i=1}^{m-1} |S_i| = \sum_{i=1}^{m} |C_i| - m$$

(1)

To determine the final structure of the union tree, we introduce the concept of potential. As part of the of the junction tree construction procedure, it is assumed the marginalization of variables, which reduces the potential definition scope. The potential is a discrete function that allows to write probability distributions in the following form $P(X_i|Y_1,Y_2,...,Y_n) = \phi(X_i,Y_1,Y_2,...,Y_n)$. The total number of potentials for a discrete BN will be determined by $[5]$

$$\delta(\phi(X_1,X_2,...,X_{k-1},X_{k+1},...,X_n)) = \prod_{X_i \in A} D_X,$$

(2)

where $D_X$ – is the number of possible values of the variable $X$.

Marginalization is the process of accumulation values for a particular variable:

$$\phi(X_1,X_2,...,X_{k-1},X_{k+1},...,X_n) = \sum_{X_k} \phi(X_1,X_2,...,X_{k-1},X_{k+1},...,X_n)$$

(3)

Under the network polling process operations of potentials projection and multiplication are also used:

$$\phi(X_1,X_2,...,X_k) = \sum_{X_{k+1},...,X_n} \phi(X_1,X_2,...,X_k,X_{k+1},...,X_n),$$

(4)

$$\phi(X_n;Y_k;Z_k) = \phi_1(X_n;Y_k) \phi_2(Y_k;Z_k)$$

(5)
On the basis of the union tree, the structure of the junction tree for the Bayesian network is determined. Then we provide that each vertex $C_i$ corresponds to all potentials of conditional probabilities, that scopes are contained in the set of vertices included in this clique: $\varphi(X_1, X_2, \ldots, X_n) \rightarrow C_i \Leftrightarrow \{X_1, X_2, \ldots, X_n\} \in C_i$. In the process of the junction tree manipulation, separators will contain potentials, that scope will be constitute as the set of separator variables – at this stage, the potentials are contained only in the clicks of the tree. After building the junction tree for analyzed DBN, we can pass directly to the network polling procedure using the constructed junction tree.

The procedure of the network polling. Structurally, the network polling algorithm can be divided into collecting messages and distributing messages throughout the JT. The message $M_q$ will be implied as a set of potential values associated with a set of JT separators $S_j$. The message collection process involves determining the potentials for each separator by applying the projection and multiplication of potentials. The message distribution is used to determine the probability distributions in each click of the junction tree due to the separator potentials obtained as a result of message collection step. For the junction tree pooling used trust propagation algorithm (Shafer-Shenoy algorithm) [6]. Underway of applying the algorithm, message is propagated in the forward and backward directions across whole edges of the junction tree. During the execution of the trust propagation algorithm, message value between two neighboring clicks $C_i$ and $C_j$ linked by separators $S_{i,j} = C_i \cap C_j$ defined as the product of all the potentials associated with the click $C_j$ in the process of accepting all messages $M$ for each click except observed click $C_j$. Note that for BN with discrete variables, the potentials for nodes can be represented as sets of conditional probability tables, and for networks with continuous variables as a Gaussian or exponential distribution. Expressions for distribution messages $M_i$ between two clicks $C_i$ and $C_j$ have the following form [7]

$$M_q(S_q) = \sum_{C_i \cap S_j} \phi_{C_i} \prod_{k \in j} M_{k_i}(S_{k_i})$$

$$P(C_i) \propto \phi_{C_i} \prod_{k \in j} M_{k_i}(S_{k_i})$$

where $M_i$ and $M_{k_i}$ – distributed messages from clique $C_i$ to $C_j$, $\phi_{C_i}$ – potential of the clique $C_i$.

The Shafer-Shenoy polling algorithm has a significant drawback – the complexity of the algorithm is proportional to the total number of neighboring vertices $n$ for the current vertex $X_i$, that’s why during the algorithm processing it is necessary to store two tables of messages values for each of the separators $S_q$ and $S_p$ both for forward and backward propagation.

To solve these problems, appropriate to use the Hugin algorithm in the network polling process. Unlike Shafer-Shenoy, this algorithm involves a single marginalization. Structurally, this algorithm is divided into two main stages: message collection and message distribution. In the process of collecting messages, selected variable $X$ that propagates information to all neighboring vertices a and recursively repeat this operation for their neighboring vertices, except the vertex $X$ itself. In this context, there is a validation only for one variable $X$, which allows to omit half messages and start spreading messages for another part. In the messages distributing process a vertex $X$ asks neighboring vertices to send message to other vertices. If this operation is implemented, then for each neighboring vertex $X'$, message is propagated to all neighboring vertices except the vertex $X$. The result of performing a network polling with the Hugin algorithm is described with the following two expressions [8]
\[
\varphi' = \sum_{\mathcal{C} \in \mathcal{S}_{i,j}} \varphi_{i,j} \quad \mathbf{M}_{i,j} = \frac{\varphi'_i}{\varphi_S} \\
\varphi_S = \frac{E_{X \rightarrow X'}}{E_{X \rightarrow X'}}
\]

where \( \varphi_{i,j} \) - potential, corresponding to the separator \( S_{i,j} \) for the clique \( \mathcal{C}_i \), \( \varphi_S \) - updated potential value for the variable \( X \), obtained due to the absorption of clicks \( \mathcal{C}_j \), \( E_{X \rightarrow X'} \) and \( E_{X \rightarrow X'} \) - messages generated during the transition from the vertex \( X \) to \( X' \).

Similar to static BN, in relation to DBN, the most effective algorithms worked out for solving learning and probabilistic inference tasks are also clustering algorithms, in particular algorithms based on the junction tree. A key feature of the JT, in relation to the semantics of the DBN, is the ability to clusterize adjacent nodes in neighboring time slices. This makes it easier to build a transition model \( P(X_{t+1} | X_s) \). The procedure for applying the JT to optimize the structure of the DBN is possible due to its sequential deployment of the network on \( k \) time slices. The main stages of JT building algorithm implementation for a static BN are presented above. In this research, we consider the procedure for adapting the JT algorithm for DBN semantic and to simplify the probabilistic inference procedure in the DBN based on a multi-particle filter (MPF). Taking into account that DBN has transitive connections, the set of clicks \( C_{t+1} \) and separators \( S_{t+1} \) can include DBN nodes located in different time slices and have conditional connections specified within the transition model \( P(X_{t+1} | X_s) \). The main idea of the proposed algorithm is to construct strongly connected trees separately for each time slice of the DBN. Consumption of clustering allows to reduce the number of node clicks \( C_{t+1} \) having temporal links between slices. Clusters are associations of vertices that have links between different time slices and defined by transition model. Accordingly, we can initially build a JT for each time slice, without taking into account the transitive connections. After that, the JT relying on nodes described by the transition model and included in the clusters. This makes it possible to work with clicks \( C_{t+1} \) that are directly defined for nodes that have temporary links. The procedure for constructing a junction tree for the DBN consists of the following main steps.

At the first step, we build a JT for each DBN time slice \( \{J_s, J_{s+1} + \ldots + J_{s+k}\} \) according to the above algorithm for building the JT for the static BN, but without implementing a network polling. As a result, we determine all possible clicks, potentials and separators for the JT for each time slice of the DBN.

The next step is to determine the a priori clusters probability distribution for the first time slice \( P(\Omega_1 | E_{t+1}) \). The cluster \( \Omega_1 \subseteq X_s \) includes all variables that have relationships in the slices \( t \) and \( t+1 \).

Whereas, we have previously constructed junction trees for each time slice \( J_{s+1} \), this distribution is quite easy to get from the potentials \( \varphi_1(X_1, X_2, \ldots X_n) \) for each clique \( \mathcal{C}_j \) of the DBN cluster in the process of building the JT up to the slice \( t + k \).

In the third step, we calculate the probability distributions for each DBN node under the slice \( t + k \) by multiplying the values from the conditional probability tables on the corresponding potentials \( \varphi_{t+1}(X_{t+1}) \) in the junction tree \( J_{t+1} \). Then define messages values for each click \( C_{t+1} \).

In the fourth step, we perform the Hugin algorithm by spreading messages starting from the root click \( C_{t+1} \). In this case, we get an update of the potentials \( \varphi_{t+1}(X_{t+1}) \) based on the expressions (8), (9).

In the next step, we get a conditional probability distribution for the cluster \( \Omega_1 \) if there is no message received up to the slice \( t + k \). Under the message we imply the clique \( \mathcal{C}_i \) and \( \mathcal{C}_j \) potentials, located in the neighbor time slices of the DBN. In this case, the conditional distribution across all clusters
$P_\text{evidence}(\Omega_t \mid E_{t+k})$ if defined evidence $E_{t+k}$, is obtained by marginalizing the potentials for each clique $C_t^i$ and $C_{t+k}^i$. Potentials $\varphi_t(\Omega^i_t)$ calculated for each time slice preceding the slice $t+k$.

In the sixth step, we update the potentials $\varphi_t(X_i)$ corresponding to the click $C_t^i$ by absorbing the click $C_{t+k}^i$ potentials based on the following expression

$$\varphi_t(X_i) = \varphi_t(X_i) \times \frac{\sum_{C_t \subseteq C_i} \varphi_{t+k}(X_{t+k})}{\sum_{C_t \subseteq C_i} \varphi_t(X_i)} \quad (10)$$

At the final stage, we repeat the Hugin algorithm by spreading $E_i$ for all clicks, starting with the click $C_t^i$. As a result, we get all the potentials for time slices. Accordingly, the total weight of the JT for the DBN under each time slice $t+k$ is determined based on the following expression

$$w(J_{t+k}) = \sum_{C_t \subseteq \{C_t^1, ..., C_t^n\}} N_i \prod_{X \in C_t} \Omega_{t+k} \quad (11)$$

where $N_i$ – total number of separators, associated with clique $C_t^i$.

A distinctive feature of the JT clustering algorithm (JTC) is that it provides mechanisms to reduce the total number of clicks for each DBN slice and the procedure for distributing messages becomes simpler and more transparent. In comparison with the Boehner-Koller algorithm [9], which also uses the junction tree as the main tool for constructing the logic of the algorithm, we do not use interfaces in the JTC algorithm. This is aligning with that in the process of forming $J_{t+k}$ the interface can get nodes not only described in the transition model, but also in the perception model. This fact significantly complicates the processing and construction DBN resulting junction tree with a large number of nodes, since it is possible to recalculate the conditional probabilities for each node in $t+k$ time slices. Another disadvantage Boyen-Koller algorithm is using of Shafer-Shenoy algorithm in the process of distribution of messages. In this case, the total complexity of the propagation algorithm will be proportional to the total number of interface nodes that make up the DBN, and the separator tables stored for each DBN slice during both forward and backward propagation. An example of the DBN and its corresponding JT for the time slice $t+1$ are shown in figure 1 and figure 2.

**Figure 1.** Original DBN.

**Figure 2.** JT for slice $t+1$ of DBN.

The procedure of building JT simplifying algorithm based on clustering, allows to reduce the total number of DBN parameters by forming clicks and clusters for all time slices of the DBN. The JTC
algorithm can be easily adapted to any algorithm for probabilistic inference based on Markov Chain Monte Carlo. This approach satisfied by the assertion that the probability distribution of JT obtained through the JTC algorithm application is fully match distribution of the original DBN. This can significantly increase productivity and reduce the complexity of probabilistic inference algorithms. The developed JTC algorithm is a special case of the JT construction algorithm, but it is more adapted to the structure and semantics of the DBN.

3. Junction tree nodes probability distribution

Let's consider calculation of probability distribution for DBN and a junction tree build upon the DBN. The complete joint probability distribution for DBN can be determined by the chain rule, as well as by setting the transition $P(X_{t+1} | X_t)$ and perception $P(E_{t+1} | X_{t+1})$ models corresponding to time slices $t$ and $t+1$ has the following form [10]

$$P(X_0, X_1, \ldots, X_t, E_1, \ldots, E_t) = P(X_0) \prod_{i=1}^{t} P(X_i | X_{i-1}) P(E_i | X_i)$$

In formula (11), we impose a number of restrictions related to the fact that the transition process between slices $t$ and $t+1$ is a Markov process of the first order [11]

$$P(X_{t+1} | X_t) = P(X_{t+1} | X_t)$$

$$P(E_i | X_{t+1}, E_{t+1}) = P(E_i | X_t)$$

Expression (14) describes the conditional independence of evidence $E_{t+1}$ relative to network variables $X_t$ when the $X_{t+1}$ variable values are known. Then, if the probability distribution $P(X_{t+1})$ will take values other than zero, then you can rewrite the expression (15) in the following form [12]

$$P(E_{t+1} | X_{t+1}, X_t) = \frac{P(E_{t+1} | X_{t+1}) P(X_{t+1} | X_t)}{P(X_{t+1})}$$

The expression (15) can be used to construct a complete probability distribution for the perception model $P(E_{t+1} | X_{t+1})$. If the probability $P(X_{t+1})$ corresponding to the distribution over all the query variables for the slice $t+1$ is zero, then based on the probability property we get

$$P(E_{t+1} | X_{t+1}, X_t) P(X_{t+1}) = P(E_{t+1} | X_{t+1}) = P(X_{t+1}) = 0$$

In this case, using expression (16), you can exclude the situation of division by zero in expression (15), since all probabilities will take zero values. To form a complete joint distribution, we assume that the conditional distributions for each JT node will be compatible with separators defined under the process of forming clicks $C_{i1}$, as well as the condition of conditional independence of such vertices [13]. The complete joint distribution for two neighbor JT nodes can be obtained from the expression (15)

$$P(E_{t+1} | X_{t+1}, X_t) = \frac{P(E_{t+1} | Parents_1(E_{t+1})) \times P(X_{t+1} | Parents_2(X_{t+1}))}{P(X_{t+1} | Parents_2(X_{t+1}))} \times P(X_{t+1} | Parents_3(X_{t+1}))$$

$$P(X_{t+1} | Parents_2(X_{t+1})) = P(X_{t+1} | Parents_3(X_{t+1})) \times P(X_t | Parents_1(X_t))$$
where $\text{Parents}_2(X_{i+1}) = \text{Parents}(X_{i+1}) \cap \{X_{i+1}^1, ..., X_{i+1}^m\}$, $\text{Parents}_3(X_i) = \text{Parents}(X_i) \cap \{X_{i+1}^m, X_i^1\}$, $\text{Parents}_4(E_{i+1}) = \text{Parents}(E_{i+1}) \cap \{E_{i+1}^1, E_{i+1}^2, ..., E_{i+1}^m, X_{i+1}^1, X_{i+1}^2, ..., X_{i+1}^m \}$ — set of parent variables for each variable $E_{i+1}$, $X_{i+1} \in X$, respectively.

Expression (17) can be simplified if we consider the entering variables $E_{i+1}$ and their parents $\text{Parents}(E_{i+1})$ in the separator $\phi_{i+1}(E_{i+1})$. Thereafter, if $E_{i+1} \cup \text{Parents}(E_{i+1}) \subset \phi_{i+1}(E_{i+1})$, then the expression (17) is reduced by $P(E_{i+1} | \text{Parents}(E_{i+1}))$. It follows that the multiplier $P(E_{i+1} | \text{Parents}(E_{i+1}))$ will take place in any cases of considering variables and their parents relative to separators $\phi_{i+1}(E_{i+1})$, and other probability distributions such as $P(E_{i+1} | W_{i+1})$ will not contribute to the complete joint probability distribution $P(E_{i+1} | X_{i+1}, X_i)$.

Similarly to the description of the probability distribution calculating procedure for the perception model $P(E_{i+1} | X_{i+1})$, you can get the initial probability distribution $P(X_0)$, as well as the transition model $P(X_{i+1} | X_i)$, if there is initial data about the separators $S_{i+1}(X_{i+1})$ obtained as a result of the JTC algorithm for the original DBN. However, we consider that as a result of the implementation of the JTC algorithm, we are limited to the assumption that DBN nodes clusters $\Omega$, formed from clicks $C_{i+1}$ only with the same adjacent nodes in neighboring time slices.

4. Conclusion

Analyzing the results of the research, we can conclude that the use of JT as an enlarged structure of the DBN allows us to significantly simplify the network topology without violating its semantic component. With provision that the semantics of the DBN is represented in the form of tables of conditional probabilities, this approach allows to exclude the fact of making any changes in the DBN probability distribution, formed through the initial distribution $P(X_0)$, the transition model $P(X_{i+1} | X_i)$ and the perception model $P(E_i | X_i)$. The JT and JTC algorithm are applied to the expanded DBN on $t + k$ slices, which requires storing each static BN associated with the specific slice. The Hugin network polling algorithm usage conditioned upon the probability distributions formation for each JT node. The JT distribution is based on determining potentials for each vertex, and updating them over the message $M_j$ distribution process. During the research, proved that the local probability distributions obtained for each of the JT nodes on each time slice are determined by similar probability distributions of the original DBN. In this case, as a result of determining the full joint probability distribution for all separators, we get that it is identical to the full joint probability distribution of the original DBN. The results of the research allow us to form the basic conditions for the construction of JT algorithms in relation to time models and, in particular, DBN. In this paper, we considered the construction of JT especially for discrete DBN, this is related with determining of values domain for each of the network variables. However, the use of DBN with continuous variables with Gaussian or exponential distributions does not impose serious restrictions on the developed JTC algorithm.

References

1. Mikaeljan S V 2011 Science and Education 10 1-25
2. Korb K B and Nicholson A E 2004 Bayesian Artificial Intelligence (New York: Chapman and Hall/CRC)
3. Jensen F V 2001 Bayesian Networks and Decision Graphs (New York: Springer-Verlag)
4. Dechter R and Pearl J 1989 Artificial Intelligence 38(3) 353–366
5. Pearl J 1988 Probabilistic Reasoning in Intelligent Systems (New York: Morgan Kaufmann).
6. Shenoy PP and Shafer G 1986 IEEE Expert 1(3) 43-52
[7] Shenoy PP 1997 *J. of Approximate Reasoning* **17** (2) 239-263
[8] Tulupyev A L 2006 *SPIIRAS Proceedings* **3** 198-227
[9] Boyen X and Koller D 1998 *UAI-98* 1 33–42
[10] Doucet A, Godsill S and Andrieu C 2000 *Statistics an Computing* **10**(3) 197-208
[11] Sirotkin A V 2006 *SPIIRAS Proceedings* **3**(1) 228-239
[12] Azarnova T V and Polukhin P V 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* **537** 052028
[13] Vyatkin A V, Filchenko A A, Tulupyev A L and Musina A L 2013 *SPIIRAS Proceedings* **3**(26) 216-233