Algebraic Bayesian Networks: Local Probabilistic-Logic Inference Machine Architecture and Set of Minimal Joint Graphs

Ekaterina A. Mal'chevskaya, Alexey I. Berezin, Andrey A. Zolotin and Alexander L. Tulupyev

Abstract The paper considers a C# software package that implements algorithms of the local probabilistic-logical inference in algebraic Bayesian networks and the synthesis of their secondary structure. The package supports local network consistency verification and maintenance, priori and posteriori inference operations. In addition, two assembly algorithms for generating the set of minimal joint graphs and a usage example for one of them are presented. These algorithms provide algebraic Bayesian networks visualisation and are intended for structured entry data representation in GUI as well as for global evidence propagation and other probabilistic-logic inference operations implementation based on their local homologues.

Keywords Algebraic Bayesian networks · Probabilistic-logic inference · Secondary structure synthesis · Assembly algorithms · Minimal joint graph

E.A. Mal'chevskaya (✉) · A.I. Berezin · A.A. Zolotin · A.L. Tulupyev
St. Petersburg State University, Saint Petersburg, Russia
e-mail: katerina.malch@gmail.com

A.I. Berezin
e-mail: beraliv.spb@gmail.com

A.A. Zolotin
e-mail: andrey.zolotin@gmail.com

A.L. Tulupyev
e-mail: alexander.tulupyev@gmail.com

E.A. Mal'chevskaya · A.I. Berezin · A.A. Zolotin · A.L. Tulupyev
St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, 39, 14-th line Vasilyevsky Ostrov, Saint Petersburg 199178, Russia

A. Abraham et al. (eds.), *Proceedings of the First International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'16)*, Advances in Intelligent Systems and Computing 451, DOI 10.1007/978-3-319-33816-3_7

1 Introduction

Increasing amounts of information and its uncertainty spur the development of mathematical models and methods to process the corresponding data volumes. Algebraic Bayesian networks (ABN) [9] are a new example of probabilistic graphical models (PGM). These models are better known with another well-known and extensively used example that is Bayesian belief networks; they serve as a representation of knowledge bases with uncertainty [2, 5, 6]. In contrast to other PGM, ABN are intended for systematic processing of both scalar and interval estimates of truth value probabilities. A probabilistic-logic inference machine is proposed over the ABN structure described previously [10], allowing both local priori and a posteriori inference in a concerned knowledge pattern [7], and similar types (or homologues) of global inference over the secondary structure [1]. In the case of global inference, the secondary structure construction time plays an important role. The purpose of this article is to consider previously obtained theoretical results on the local probabilistic-logic inference and to present the architecture of the new software complex implementing these results as well as minimum joint graph synthesis algorithms.

2 Probabilistic-Logic Inference

2.1 The Tasks of Probabilistic-Logic Inference in ABN

An ABN can be represented as an undirected graph with ideals of conjuncts associated with the graph vertices. Conjuncts like other propositional formulae are defined over a fixed alphabet, at the same time scalar or interval probability estimate is attributed to every conjunct. Let us call an ideal of conjuncts with probability estimates a knowledge pattern (KP). Note that a knowledge pattern can be constructed not only over the ideal of conjuncts, but also over the ideal of disjuncts or quanta propositions [8].

There are three tasks of local probabilistic-logic inference (P-LI) in ABN: the local priori inference, the first and the second tasks of local posteriori inference. Next, let us consider the tasks in detail, as well as introduce the basic definitions of KP consistency and reconcilability [7].

The conjunct is the conjunction of some number of atomic variables of the form $x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_k}$.

Quantum over the alphabet $A = \{x_i\}_{i=0}^{n-1}$ is a conjunction which contains either the atoms or its negation for every atom of the alphabet.

Suppose given a KP with scalar probability estimates (C, p). We say that it is *consistent* and denote it by Consistent$[(C, p)]$ iff there is a probability p_F, defined over a set of propositional formulae $F(A)$ (where A is the alphabet, which the KP is built over) such that

$$\forall c \in C : p_F = p(c)$$

Suppose given a KP with interval probability estimates $(C, \mathbf{p})$. We say that it is *consistent*, and denote it as Consistent$[(C, \mathbf{p})]$ iff for every conjunct $c \in C$ and every $\varepsilon \in \mathbf{p}(c)$ there is a function $p_{c,\varepsilon} : C \to [0, 1]$ such that $p_{c,\varepsilon} = \varepsilon$ and $(C, p_{c,\varepsilon})$ is consistent in terms of previous definition.

Suppose given a KP with interval probability estimates $(C, \mathbf{p})$. We say that it is *reconcilable* iff there is a consistent KP with interval probability estimates $(C, \mathbf{p}')$ such that

$$\forall c \in C : \mathbf{p}'(c) \subseteq \mathbf{p}(c)$$

The local priori inference task is to build the truth values of propositional formulae defined over the same alphabet as a KP based on a consistent KP.

Before giving a definition of evidence propagation tasks let us represent the concept of evidence used in these tasks.

Evidence is new "conditional" data that arrived in KP, and taking that into account we need to reconsider all (or only some) probability estimates. The evidence is represented as $\langle \cdots \rangle$.

The first task of the posterior inference is to estimate a probability of the evidence, given KP elements probability estimates.

The second task of the posterior inference is to evaluate the conditional probabilities of KP elements assuming that an evidence has arrived.

The theory of ABN considers 3 types of evidences: deterministic, stochastic and interval.

Deterministic evidence is the assumption that a conjunction of atoms or negotiation of atoms appears to be truth. This evidence is denoted by $\langle c_i, c_j \rangle$. Here c_i, c_j are the conjuncts which composed of atoms(without negation) and the negation of atoms respectively.

Stochastic evidence is the assumption that consistent KP with scalar probability estimates that determine the corresponding subideal elements probabilities is set over C' which is the subideal of C. This evidence is denoted by $\langle (C', \mathbf{P_c}) \rangle$.

Interval evidence is the assumption that consistent KP with interval probability estimates which determine the corresponding subideal elements probabilities is set over C' which is the subideal of C. This evidence is denoted by $\langle (C', \mathbf{P_c^-}, \mathbf{P_c^+}) \rangle$.

All the tasks above form a local probabilistic-logic inference in ABN, which was implemented in the software package further described.

2.2 Software Package Architecture

Designing the architecture of the software package, we paid a particular attention to the structure of the classes and interfaces that are responsible for the creation, storage and processing of KP. Three types of KP are considered in software

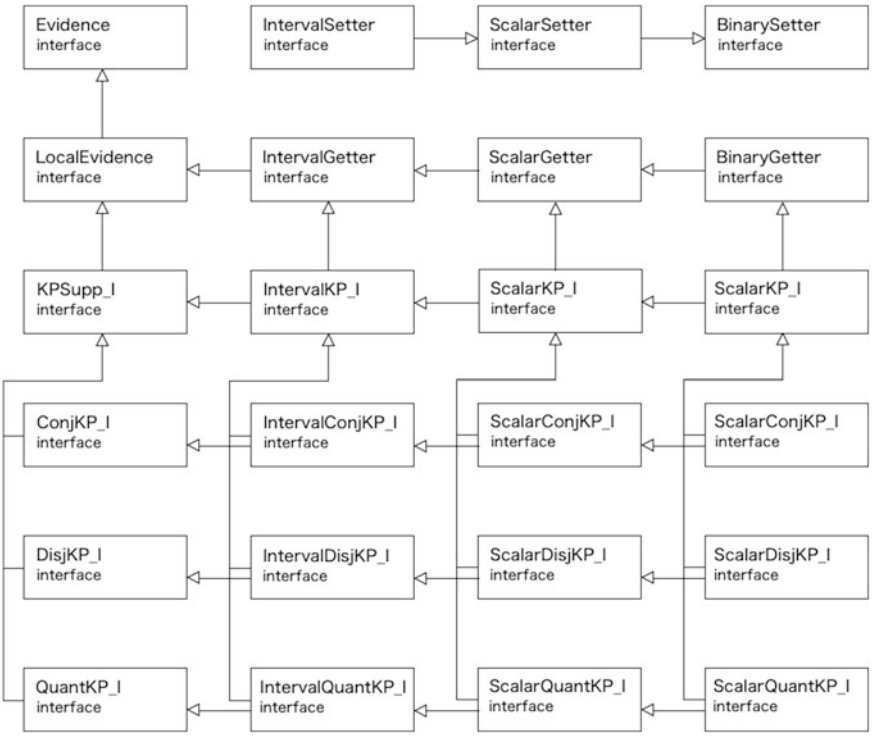

Fig. 1 Interfaces diagram

package: interval, scalar and binary. It has been observed that a binary KP is a particular case of a scalar KP and a scalar KP is a particular case of the interval KP. The observation gave rise to two inheritance chains. The first chain is the inheritance of getters, methods that read KP probability estimates, where a binary KP might be read as a scalar KP while scalar KP might be read as an interval KP. The second chain is the inheritance of setters, methods that set the KP probability estimates. The structure of interfaces interconnections which aggregates observations mentioned above, shown in Fig. 1.

The software package has a separate inferrers structure, responsible for the KP consistency checking and maintainance. It also lets convert a KP over an ideal of conjuncts into a KP over quanta propositions. The Fig. 2 shows the inferrers structure classes and interfaces dependencies.

The propagators structure in the software package is responsible for the local posteriori P-LI, that is to solve the first and second tasks of the evidence propagation. Methods described in the inferrers structure are used to maintain the consistency of the KP after evidence is propagated, corresponded to the DRY (Do not Repeat Yourself) principle. The Fig. 3 shows the dependence of classes and interfaces, responsible for the deterministic evidences propagation from each other.

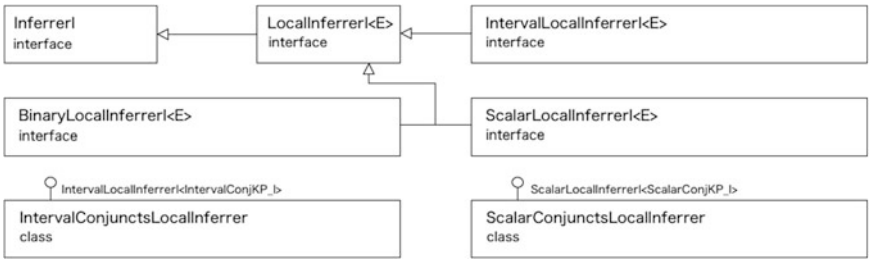

Fig. 2 Inferrers structure

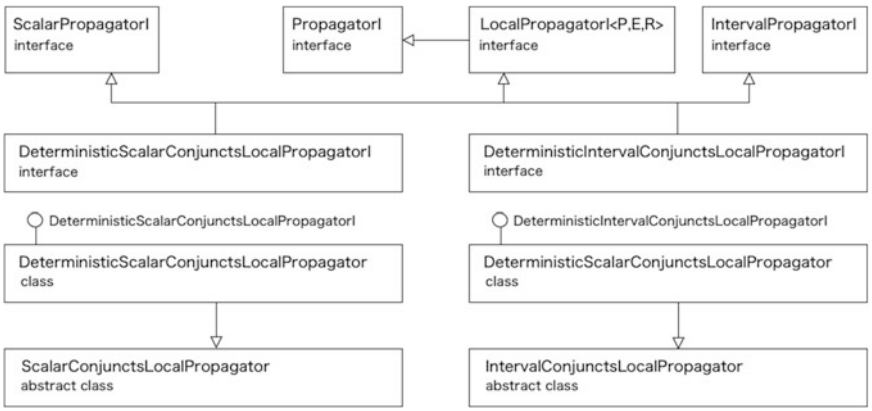

Fig. 3 Propagators structure

2.3 Probabilistic-Logic Inference Implementation

The developed software package allows to construct knowledge patterns, which can be built over the ideals of conjuncts, disjuncts or quanta propositions, with different types of probability estimates: binary, scalar or interval and conduct a P-LI over the derived KP.

As an example of the P-LI let us consider maintenance of consistency in the knowledge pattern, which is built over the ideal of conjuncts with interval probability estimates.

Let us consider the KP over the alphabet consisting of two symbols:

$A = \{x_1, x_2\}$. Let's take probability estimates of conjuncts:

$$\left(\mathbf{P}_{\mathbf{c}}^{-}\right)^{\mathrm{T}} = (0 \quad 0 \quad 0.2 \quad 0.2) \text{ and } \left(\mathbf{P}_{\mathbf{c}}^{+}\right)^{\mathrm{T}} = (1 \quad 1 \quad 0.7 \quad 0.7).$$

As a result of their adjustment with the methods of the class Interval- Conjuncts-Local- Inferrer we receive the detailed probability estimates:

$$\left(\mathbf{P}_{\mathbf{c}}^{-}\right)^{\mathrm{T}} = (0.2 \quad 0.2 \quad 0.2 \quad 0.2) \text{ and } \left(\mathbf{P}_{\mathbf{c}}^{+}\right)^{\mathrm{T}} = (1 \quad 1 \quad 0.7 \quad 0.7).$$

It is obvious that the estimates are narrowed, and verification confirms the consistency of the obtained KP.

3 SMJG

The set of minimal joint graphs (MJG) is considered in the section, assembly algorithms for set synthesis are presented, which then allow to select the optimal one for probabilistic-logic inference.

3.1 Necessity of Use Secondary Global Structure

Having considered ABN KP representations and local P-LI algorithms implementations [11], it is impossible not to mention the usage context of the upcoming C# library.

At first, note should be taken that researchers mostly cannot limit themselves with processing a single KP: the tasks they fulfil in the main include set of KP representations and processing, i.e. ABN. This requires to bring following issues to a close: adding information about set of KP and having a convenient secondary global structure visualisation, to represent links among KPs.

Besides, built library includes global P-LI algorithms in ABN. Therefore, it requires to settle a problem of ABN secondary global structure synthesis in order to provide dissemination of impact on local changes across the network, propagation of evidence, verification of a variety of ABN consistency degrees.

The current approach in the ABN theory is that the secondary global structure is MJG, though these several graphs can be corresponded to same set of KP. For this reason, we consider structural tasks in details paying more attention to MJG and assembly models [12].

3.2 Definitions

A primary structure is a set which should consist of inclusion maximal subalphabets: $\forall w_i, w_j \in$ Workloads, $i \neq j \Rightarrow w_i \subsetneq w_j$. A collection of these sets is denoted by PrimaryStructures.

Compression $\sigma_U : G \to G$ is a map $\sigma_U(G) = K_U$, where $K_U = \langle F_U, E_U, d(e)\rangle$, such that

1. $F_U = \{f_i | f_i = \sigma_U(P_U^i), P_U^i \subseteq V(G \downarrow U)\};;$
2. $E_U = \{e_i | e_i = \sigma_U(E_{i,j}), E_{i,j} \subset (G \downarrow U) \setminus (G \Downarrow U)\};;$
3. $d : E_U \to N : d(\sigma_U(E_{ij})) = |E_{ij}|$, where $|E_{ij}|$ is the multiplicity.

A feud of a weight U, or f_i, is a vertex obtained by compressing some possession. *A curia* of the weight U, or K_U, is a multigraph which came out from compressing the narrowing $G \downarrow U$. *A homage* H_U is a curia K_U being a tree in which all the edges have multiplicity 1.

A sinew $s_{i,j}^U$ is a graph, built on vertices of the restriction $M \downarrow U$ and the corresponding homage $H_i^U : \sigma_U(s_{i,j}^U) = H_i^U$. The set of sinews of the weight U is denoted by Sinews_U.

A sheaf is a graph, constructed by unification of sinews, selected by one for every separator:

$$\bigcup_{U_k \in Sep} s_{i_k,j_k}^{U_k}, \ where \ s_{i_k,j_k}^{U_k} \in Sinews_{U_k}.$$

3.3 MJG and Assembly Algorithms

Next, the matter will be subsets of MJG synthesis algorithms, where subsets can be either the whole set of MJG (SMJG) or a singleton.

Denoted ModelKit as the following quadruple ⟨Loads, StereoSeparators, StereoHoldings, NecessaryEdges⟩, where Loads is a primary structure, StereoHoldings is a set of strong narrowing components, where narrowing influences every stereoseparator:

$$\text{StereoHoldings} = \{\text{Components}(\downarrow U) | \ U \in \text{StereoSeparators}\},$$

where Components returns set of graph connected components.

NecessaryEdges is a set of necessary edges:

$$\text{NecessaryEdges} = \bigcup_{U \in \text{BSep}} E(\downarrow U),$$

where BSep is a set of biseparators.

MJG subsets synthesis algorithms given ModelKit are named *assembly algorithms*.

Next, we will see several SMJG synthesis algorithms. The efficiency of the suggested algorithms is mentioned in [3].

3.4 MK1 Algorithm Description

In listing 1 there is a pseudocode of SMJG synthesis algorithm MK1.

Algorithm 1 Simple algorithm generating SMJG MK1

input: Loads
output: ModelKit

```
 1: StereoSeparators = ∅
 2: StereoHoldings = ∅
 3: NecessaryEdges = ∅
 4: ⟨ Separators, SeparatorTable ⟩ = SeparatorSet(Loads)
 5: for each s in Separators
 6:     Vertices = NarrowedVertices(s, Loads)
 7:     if |Vertices| = 2 then
 8:         NecessaryEdges = NecessaryEdges ∪ {Vertices[0], Vertices[1]}
 9:     else
10:         Holdings = Components(StrongNarrowing(Vertices, s, SeparatorTable))
11:         if |Holdings| > 1 then
12:             StereoSeperators = StereoSeperators ∪ {s}
13:             StereoHoldings[s] = Holdings
14: return ⟨ Loads, StereoSeparators, StereoHoldings, NecessaryEdges ⟩
```

The idea of MK1 is following: first, the set of non-empty separators is constructed (line 4), second, the set of vertices which are from respective narrowing is constructed for every separator going over all loads (line 6). Third, it is checked for the constructed set of vertices if the separator is a biseparator (i.e. the number of vertices equals 2) or a stereoseparator (i.e. it is not a biseparator and the number of possessions are greater than 1). In first case (lines 7–8) the edge is added to the set of necessary edges. In second case the possessions are constructed as the connected components of strong narrowing (lines 10–13) [3] (Fig. 4).

3.5 MK4 Algorithm Description

In listing 2 there cites a pseudocode of SMJG synthesis algorithms MK4.

Algorithm 2 Advanced SMJG synthesis algorithm MK4

input: Loads
output: ModelKit

```
 1: StereoSeparators = ∅
 2: StereoHoldings = ∅
 3: NecessaryEdges = ∅
 4: ⟨ Separators, MainVertices, SeparatorTable ⟩ =
 5:       SeparatorSetAndMainVertices(Loads)
 6: Sons = Sons(Separators)
 7: SortedSeparators = SortSeparators(Separators)
 8: for each s in SortedSeparators
 9:       Vertices = MainVertices[s]
10:       if |Vertices| = 2 then
11:             if Sons[s] = ∅ then
12:                   NecessaryEdges = NecessaryEdges ∪ {Vertices[0], Vertices[1]}
13:             else
14:                   for each c in Sons[s]
15:                         Vertices = Vertices ∪ Vertices(c)
16:       else
17:             Holdings = HoldingsBy4thStructure(s,Sons,Vertices)
18:             if |Holdings| > 1 then
19:                   StereoSeperators = StereoSeperators ∪ {s}
20:                   StereoHoldings[s] = Holdings
21:             else
22:                   for each c in Sons[s]
23:                         Vertices = Vertices ∪ Vertices(c)
24: return ⟨ Loads, StereoSeparators, StereoHoldings, NecessaryEdges ⟩
```

The idea of MK4 is following: at first a set of non-empty separators is constructed simultaneously with the set of main vertices of the separators (lines 4–5), then the tertiary structure is constructed (line 6). Separators are considered ascending (line 7), if the separator *s* has two main vertices and has no children, it is called a biseparator and processed accordingly (lines 10–12). If it has children, it is called a monoseparator of type 1, and then the set of vertices is constructed joining the sets of his sons vertices with its own main vertices (lines 14–15). If the separator has more than two main vertices, the set of his possessions is synthesised constructing a half-sibling graph and finding the connected components in it. If the number of possessions is greater than 1, then the separator is processed as the stereoseparator (lines 18–20). If the number of possessions equals 1, the separator is a monoseparator but not of type 1, and a set of his vertices is constructed same way as for a monoseparator of type 1 (lines 22–23). That both monoseparators' types processing is necessary that while constructing the monoseparators' parent possessions all the monoseparators' vertices are taken into account but not only those which the main one are [4].

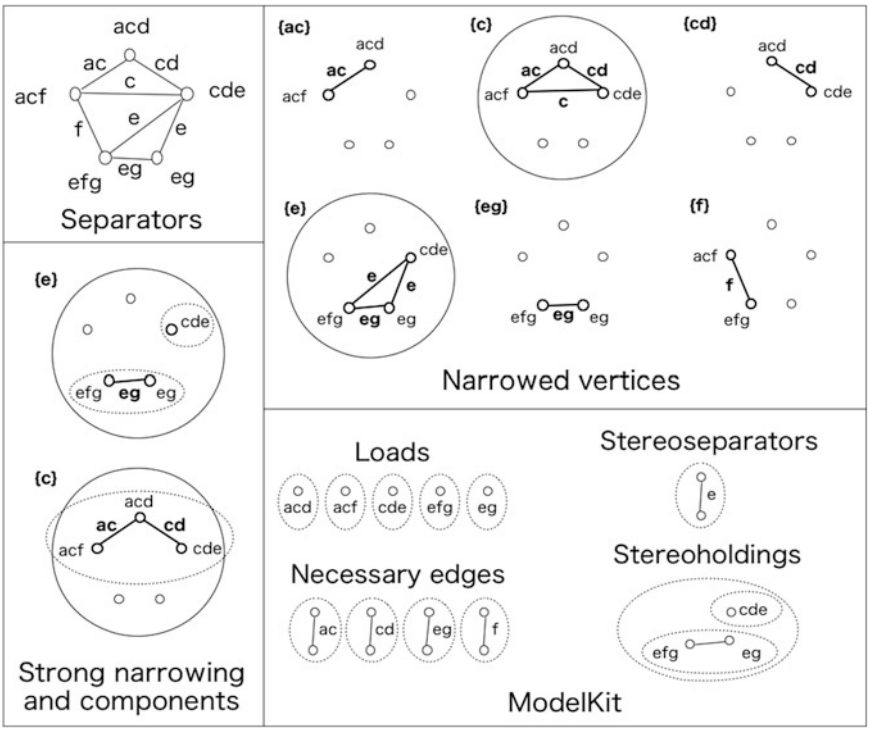

Fig. 4 The work example of MK1

3.6 The Work Instance of Assembly Algorithms

Similarly, the example of the algorithm MK4 could be referred to.

4 Conclusion

Thus, we considered the primary and the secondary structure of the ABN. For the primary structure we introduced the basic terms and concepts relating to the PLI and described the developed C# library implementing all PLI algorithms using side library lp_solve for linear programming problems solving. Designed library corresponds to the basic principles of object-oriented programming, which makes it expandable and has API, making it reusable in future global PLI algorithms implementation. Also we described 2 algorithms for minimal joint graph generation, supplemented by a program code listings and examples of their use. The obtained secondary structure gives an opportunity to extend application of local inference machine so that it can conduct a variety of global probability-logic inference types.

Acknowledgments The paper presents results of the project partially supported with RFBR grant 15-01-09001-a "Combined Probabilistic-Logic Graphical Approach to Representation and Processing of Unsertain Knowledge Systems: Algebraical Bayesian Networks and Related Models".

References

1. Butz, C.J., Oliveira, J.S., Madsen, A.L.: Bayesian network inference using marginal trees. Int. J. Approx. Reason. **68**, 127–152 (2015)
2. Farasat, A., Nikolaev, A., Srihari, S.N., Blair, R.H.: Probabilistic graphical models in modern social network analysis. Soc. Netw. Anal. Min. **5**(1) (2015)
3. Filchenkov, A.A., Frolenkov, K.V., Sirotkin, A.V., Tulupyev, A.L.: Minimal joint graph subsets synthesis system. Tr. SPIIRAN **27**, 200–244 (2013). http://mi.mathnet.ru/trspy656 (in Russian)
4. Fil'chenkov, A.A., Tulupyev, A.L.: Minimal joint graph structure synthesis. Tr. SPIIRAN **11,** 104–129 (2009). http://mi.mathnet.ru/trspy50 (in Russian)
5. Kappel, D., Habenschuss, S., Legenstein, R., Maass, W.: Network plasticity as Bayesian inference. PLoS Comput. Biol. **11**(11) (2015)
6. Nelson, A.W., Malik, A.S., Wendel, J.C., Zipf, M.E.: Probabilistic force prediction in cold sheet rolling by Bayesian inference. J. Manuf. Sci. Eng.-Trans. ASME **136** (2014)
7. Sirotkin, A.V.: Algebraic Bayesian networks: computational complexity of algorithms for probabilistic-logical inference under uncertainty (2011) (in Russian)
8. Tulupyev, A.L., Nikolenko, S.I., Sirotkin, A.V.: Bayesian networks: a probabilistic-logic approach. SPb.: Nauka (2006) (in Russian)
9. Tulupyev, A.L., Sirotkin, A.V., Nikolenko, S.I.: Bayesian belief networks. SPb.: SPbSU Press (2009) (in Russian)
10. Tulupyev, A.L., Sirotkin, A.V., Zolotin, A.A.: Matrix equations in a posteriori inference of truth estimates in algebraic Bayesian networks. Vestnik St. Petersburg University. Mathematics **48**(3), 168–174 (2015)
11. Tulupyev, A.L., Stolyarov, D.M., Mentyukov, M.V.: A representation for local and global structures of an algebraical Bayesian network in java applications. Tr. SPIIRAN **5**, 71–99 (2007). http://mi.mathnet.ru/trspy301 (in Russian)
12. Zotov, M.A., Tulupyev, A.L., Sirotkin, A.V.: Complexity statistical estimates of straightforward and greedy algorithms for algebraic Bayesian networks syntesis. Nechetkie Sistemy i Myagkie Vychisleniya [Fuzzy Systems and Soft Computing] **10**(1), 75–91 (2015) (in Russian)