СИНТЕЗ ГРАФА СМЕЖНОСТИ С МИНИМАЛЬНЫМ ЧИСЛОМ РЕБЕР: ФОРМАЛИЗАЦИЯ АЛГОРИТМА И АНАЛИЗ ЕГО КОРРЕКТНОСТИ

Опарин В.В., Тулупьев А.Л.

УДК 004.8

Опарин В.В., Тулупьев А.Л. **Синтез графа смежности с минимальным числом ребер: формализация алгоритма и анализ его корректности.**
Аннотация. Предлагается алгоритм формирования вторичной структуры алгебраической байесовской сети (АБС) на основе ее первичной структуры. Вторичная структура АБС представляет собой граф смежности с минимальным числом ребер. Приведено доказательство корректности работы алгоритма.
Ключевые слова: алгоритм, алгебраическая байесовская сеть, граф смежности, универсальное множество нагрузок, база фрагментов знаний.

Oparin V.V., Tulupyev A.L. **Synthesis of a joint graph with minimal number of edges: an algorithm formalization and a proof of correctness.**
Abstract. The paper describes an algorithm for synthesis of the secondary structure of an algebraic Bayesian network (ABN) by its first structure. The second structure of ABN is a joint graph with minimal number of edges. A proof of algorithm's correctness is given.
Keywords: algorithm, algebraic Bayesian network, joint graph, universal loads set, knowledge patterns base.

1. Введение. Дефицит знаний (knowledge bottleneck), сдерживающий развитие и промышленное внедрение интеллектуальных систем, может быть преодолен, в частности, за счет развития теории и, затем, технологий машинного обучения (machine learning) формальных моделей знаний с неопределенностью[1] [14, 17]. Алгебраические байесовские сети (АБС) являются одной из таких моделей или, более точно, они относятся к классу логико-вероятностных графических моделей баз фрагментов знаний с неопределенностью [11].

[1] В ряде научных областей используются близкий по значению термин — синтез моделей. Он осуществляется, как и машинное обучение, на основе некоторого набора исходных данных и знаний, которые могут отличаться неполнотой, неточностью и/или нечеткостью.

Проблема машинного обучения АБС распадается на две основные составляющие: машинное обучение фрагмента знаний (локальное обучение, идентификация численных параметров — оценок вероятностей истинности конъюнктов, которые являются элементами фрагмента знаний) и машинное обучение глобальной структуры алгебраической байесовской сети (структурный синтез, синтез структуры, идентификация структуры модели). В теоретическом и алгоритмическом аппарате обеих составляющих машинного обучения алгебраических байесовских сетей, а также представления и обработки последних был сформирован заметный задел [1–15], однако он требует развития, поскольку остается нерешенным спектр задач, возникающих при попытке развития соответствующего комплекса программ и реляционных баз данных.

Цель настоящей работы состоит в описании алгоритма построения минимального (по числу ребер) графа смежности над заданным множеством фрагментов знаний и обосновании корректности работу указанного алгоритма. Указанный алгоритм нацелен на формирование одного отдельного графа смежности. Исследование базовых свойств системы минимальных графов смежности, построенных для алгебраической байесовской сети с заданной первичной структурой нашло освещение в работе [16].

Представление вторичной структуры АБС (или, что можно рассматривать как синоним, глобальной структуры АБС) в виде графа смежности впервые было предложено в [12]. Затем система определений и применение графов смежности (и их подвидов) были развиты в ряде публикаций, в частности, в работах [1–3, 5, 6, 13–15]. Исходные результаты по автоматизации формирования графов смежности представлены в [14]; кроме того, С.С. Синчук и М.А. Левин реализовали программные приложения, осуществляющие синтез графов смежности, однако свои результаты не опубликовали. Применение графов смежности и их подвидов оказало существенное влияние на разработку структуры данных в коде и реляционных базах данных, а также на реализацию алгоритмов глобального логико-вероятностного вывода в комплексе программ, описанном в работах [1, 2, 11].

2. Постановка задачи. Задан конечный индексированный алфавит

$$A = (x_1, x_2, ..., x_m)$$

и конечное множество вершин

$$V = \{v_1, v_2, ..., v_n\}.$$

Задано отображение $W : V \to 2^A$, сопоставляющее каждой вершине $v \in V$ некоторое множество $W(v) \subset A$. Множество $W(v)$, обозначемое далее W_v, назовем *весом вершины* v, а саму функцию W — *весовой функцией*. В более общем случае любое подмножество алфавита A будем называть *весом*.

Известно, что для любых двух различных вершин $u, v \in V$ веса W_u и W_v не являются включениями одного в другое.

Определение 1. Назовем неориентированный граф $G = \langle V, E \rangle$ *графом смежности*, если он удовлетворяет следующему условию:

> $\forall u, v \in V$ таких, что $W_u \cap W_v \neq \emptyset$, существует некоторый путь P в графе G такой, что для каждой вершины $s \in P$ справедливо утверждение $W_u \cap W_v \subseteq W_s$.

Требуется построить граф смежности $G_{\min}$ с минимальным числом ребер (далее —*минимальный граф смежности*).

3. Обозначения. Введем некоторые обозначения, употребляемые далее по тексту.

Построим порядок на множестве весов. Для удобств записи введем индексную функцию $I : A \to [1, m]$, сопоставляющую каждому символу x_i алфавита A его индекс i; иначе говоря, $I(x_i) = i$. Сопоставим каждому весу кортеж, составленный из всех элементов данного веса, такой, что если позиция одного элемента кортежа меньше другого, то и индекс первого меньше индекса второго. Более строго

$$F(W) = (a_1, a_2, ..., a_{|W|}),$$
$$\forall a_i, a_j \in W : i < j \Leftrightarrow I(a_i) < I(a_j).$$

Кортеж $F(W)$ назовем *записью* веса W.

Заметим, что благодаря введенной индексации мы можем уже говорить о лексикографическом порядке над множеством записей весов.

Назовем *мощностью веса* $|W|$ мощность множества, которое представляет данный вес. Вес W_1 будем считать меньше веса W_2 тогда и только тогда, когда либо $|W_1| < |W_2|$, либо $|W_1| = |W_2|$ и $F(W_1)$ меньше $F(W_2)$ лексикографически; иначе говоря, при

$$F(W_1) = (a_1, a_2, ..., a_{|W_1|}),$$
$$F(W_2) = (b_1, b_2, ..., b_{|W_2|})$$

справедливо, что

$$W_1 < W_2 \Leftrightarrow \exists i : (I(a_i) < I(b_i))\&(\forall j < i : I(a_j) = I(b_j)).$$

Обратим внимание, что длины кортежей в данном случае считаются одинаковыми, так как лексикографическое сравнение происходит только при условии $|W_1| = |W_2|$.

В дальнейшем свойства линейной упорядоченности множества весов не используется, и нам хватит лишь частичного порядка, задаваемого мощностями. Поэтому лексикографическое сравнение при равенстве мощностей, вообще говоря, может быть заменено на любое другое или даже отсутствовать.

Определение 2. Назовем $\Omega(V, W)$ универсальным множеством нагрузок, если

$$\Omega(V, W) = \{q \subseteq A | \exists s, t \in V : s \neq t, q = W(s) \cap W(t)\} \backslash \{\emptyset\}.$$

Для краткости обозначим $\Omega(V, W)$ как Ω. Особо отметим, что пустое множество не входит в Ω.

Условимся считать $V(G)$ и $E(G)$ соответственно множествами вершин и ребер графа $G = \langle V, E \rangle$.

Определение 3. *Нагрузкой* ребра $(u, v) \in E$ назовем вес

$$W_{(u,v)} = W_u \cap W_v.$$

Обозначим подграф графа G, индуцированный весом q, как

$$G_q = \langle V_q, E_q \rangle,$$

где

$$V_q = \{v \in V | q \subseteq W_v\},$$
$$E_q = \{(u, v) \in E | u, v \in V_q\}.$$

Для графов с дополнительными индексами вида G^m подграфы, индуцированные весом q, будем обозначать как G^m_q.

Определение 4. Конечную последовательность вершин

$$P = (p_1, p_2, \ldots, p_n),$$

где $n \geq 1$, графа $G = \langle V, E \rangle$ назовем *путем из* p_1 *в* p_n, если

$$\forall i < n : (p_i, p_{i+1}) \in E.$$

Далее путь P из вершины s в t обозначим как

$$P : s \rightsquigarrow t.$$

Будем говорить, что ребро $e = (u, v)$ принадлежит пути P, если

$$\exists i < n : \quad u = p_i \quad \& \quad v = p_{i+1}.$$

Компонентой связности (далее КС) графа $G = \langle V, E \rangle$ назовем такое множество вершин $K \subseteq V$, что для всякой пары $s, t \in K$ существует путь в графе G из s в t. Более формально

$$K = \{s \in V | \forall t \in K \backslash \{s\} \; \exists P : s \rightsquigarrow t\}.$$

Обратим внимание: в нашей терминологии компонента связности является только множеством вершин некоторого графа, но не его подграфом.

Компоненту связности в графе G_q, содержащую в себе вершину v, будем обозначать K_{vq}.

4. Алгоритм. Ниже мы приведем алгоритм нахождения минимального графа смежности, но прежде введем дополнительные обозначения.

Мы предполагаем, что Q есть *очередь с приоритетами*[2] [18]. В общем случае структура Q может быть заменена любой другой структурой, являющейся коллекцией. Элементами структуры Q будут веса из универсального множества нагрузок Ω.

Структура S предполагает некоторую реализацию множества. Она может быть как системой непересекающихся множеств, так и бинарным вектором. Общее требование: необходимы свойства коллекции. В качестве элементов S содержит вершины из множества V.

[2]Под очередью с приоритетами мы понимаем структуру данных, хранящую элементы некоторого частично упорядоченного множества. Очередь с приоритетами предоставляет быстрый доступ и удаление максимального элемента, а также добавление произвольного элемента в очередь. Как правило, очередь с приоритетами реализуют на основе кучи [18]. Если в очереди есть несколько элементов с одинаковыми максимальными приоритетами, результат операции извлечения и просмотра зависит от конкретной реализации. В нашем случае порядок следования максимальных элементов не важен.

Ввиду своих возможностей описанная структура данных представляется наиболее выразительным вариантом.

Функция delegate(S) возвращает представителя множества S, являющегося его элементом. В нашем алгоритме выбор конкретного представителя не играет ключевой роли. Модификации могут потребовать уточнений по способу выбора представителя. В настоящей работе выбор представителя в функции delegate(S) считается произвольным.

Функция component(G, v, q) возвращает КС подграфа G_q, представленную в виде множества вершин и содержащую в себе вершину v. В случе, если $q \not\subseteq W_v$, функция component(G, v, q) возвращает пустое множество.

Реализация структуры графа требует возможности добавления ребер и их перебора.

Через стрелки $\leftarrow$ соответственно будем обозначать добавление или удаление элемента из структуры:

$$\begin{aligned} A &\leftarrow a, \\ a &\leftarrow A. \end{aligned}$$

В случае упорядоченных структур, таких как очередь с приоритетом, запись $a \leftarrow Q$ будет обозначать извлечение максимального элемента Q из очереди и помещение его значения в переменную a.

На листинге 1 приведен алгоритм построения минимального графа смежности (АПМГС). Ниже мы подробно рассмотрим шаги данного алгоритма.

В строках (1—2) АПМГС инициализируются начальные структуры данных. Очередь с приоритетами Q создается пустой. Граф G создается на базе тех же вершин, что были поданы на вход АПМГС, но пока с пустым множеством ребер.

В строках (3—7) очередь Q заполняется элементами универсального множества нагрузок Ω для заданного V и W. Цикл (3—7) перебирает все пары различных вершин из множества V. В строке (4) проверяется принадлежность нагрузки $W_{(uv)}$ между вершинами u и v множеству Ω. Если $W_{(uv)}$ еще не лежит в очереди Q, то алгоритм добавляет нагрузку в Q. Последнее написано в строке 5.

В строках (8—22) представлено основное действие алгоритма. Это цикл, перебирающий нагрузки, записанные в Q. В строке $9-10$ алгоритм извлекает нагрузку q с максимальной мощностью из очереди и создает для нее пустое множество S. Далее в это множество будут складываться все вершины из графа G_q, уже соединенные в одну КС. Цикл (11—21) занимается непосредственно заполнением

Листинг. 1 Алгоритм построения минимального графа смежности

Require: V, W
Ensure: $G = \langle V, E \rangle$
1: $Q = \emptyset$
2: $G = \langle V, \emptyset \rangle$
3: **for all** $u, v \in V, u \neq v$ **do**
4: **if** $(W_u \cap W_v \neq \emptyset \ \& \ W_u \cap W_v \notin Q)$ **then**
5: $Q \leftarrow W_u \cap W_v$
6: **end if**
7: **end for**
8: **while** $Q \neq \emptyset$ **do**
9: $q \leftarrow Q$
10: $S = \emptyset$
11: **for all** $v \in V$ **do**
12: **if** $(q \subset W_v \ \& \ v \notin S)$ **then**
13: **if** $S \neq \emptyset$ **then**
14: $d = delegate(S)$
15: $S = S \cup component(G, v, q)$
16: $G.E \leftarrow (d, v)$
17: **else**
18: $S = component(G, v, q)$
19: **end if**
20: **end if**
21: **end for**
22: **end while**
23: **return** G

структуры S и добавлением необходимых ребер в строящийся граф смежности G. Цикл перебирает все вершины из графа G_q, как это указано в строках (11—12). Если некоторая вершина v оказалась за пределами множества S, алгоритм добавляет вершины КС K_{vq} в графе G_q в множество S.

При этом, возможны два варианта.

- S — пустое множество. Тогда алгоритм вызывает процедуру получения множества вершин K_{vq} и объединяет его с S. Этот случай описан в строке 18.

- S уже содержит некоторую компоненту связности. Тогда алгоритм получает некоторую вершину–представителя s из КС S, объединяет множества вершин K_{vq} с S. И добавляет ребро (sv) в граф смежности G. Данный случай представлен в строках (14—16).

Обратим внимание: на протяжении работы алгоритма множество S всегда является некоторой КС в графе G.

В строке (23) алгоритм возвращает построенный граф смежности.

Таким образом, АПМГС заполняет Q множеством нагрузок из Ω. Затем, перебирая от максимальной к минимальной каждую нагрузку $q \in \Omega$, алгоритм дополняет подграфы G_q ребрами до компонент связности.

В следующем разделе покажем, почему АПМГС строит граф смежности G, а также почему число ребер в графе G минимально.

5. Обоснования. Докажем, что АПМГС строит граф смежности. Затем докажем минимальность построенного графа. Для обозначения различных фрагментов алгоритма мы будем ссылаться на строки. Так циклом (8—22) на листинге 1 обозначим цикл *while* $Q \neq \emptyset$ *do* вместе с его телом.

Утверждение 1. *По завершении каждой итерации цикла (8—22) по* $q \in Q$ *подграф* $G_q = \langle V_q, E_q \rangle$ *становится связным.*

Доказательство. Заметим, если операцию по объединению двух КС, выполняемую в строках (15—16) , считать атомарной, то в любой момент времени S является некоторой КС (возможно пустой).

Действительно, изначально $S = \emptyset$. Функция component(G, v, q) возвращает КС K_{vq} в подграфе G_q с вершиной v. При появлении первой вершины v из V_q, множество S станет равным K_{vq}, т.е. КС.

Пусть S уже не пусто. Для текущей вершины $v \in V_q$ получена КС K_{vq}. Функция delegate(S) возвращает некоторую вершину d из множества S. При объединении S и K_{vq} в множество ребер графа G добавляется ребро (d, v). Так как и S и K_{vq} КС, то с добавлением ребра (d, v) множество вершин $S \cup K_{vq}$ также станет КС.

Цикл (11—21) пытается добавить все вершины графа G в S. Значит, по завершении его работы множество вершин граф G_q и S будут эквивалентны. Ввиду свойства S всегда представлять собой КС, получаем связность графа G_q.

Утверждение 2. *Граф G является графом смежности тогда и только тогда, когда для любого веса $q \in \Omega$ подграф G_q связен.*

Доказательство.

Рассмотрим две произвольные различные вершины s и t из графа G. Пусть $W_s \cap W_t = q \neq \emptyset$. $q \in \Omega$ по определению. Рассмотрим граф G_q. По условию он связен. $s, t \in G_q \Rightarrow$ существует путь P в G_q (а значит и в G) из s в t. Т.к. каждая вершина $p \in P$ лежит в G_q, то $q \subset W_p$. Но $q = W_s \cap W_t$. Ввиду произвольности вершин s и t граф G является графом смежности.

Рассмотрим произвольную нагрузку $q \in \Omega$. Рассмотрим две произвольные различные вершины s, t из графа G_q. Заметим, в графе G существует путь P из s в t такой, что $\forall p \in P : W_s \cap W_t \subseteq W_p$. Значит, из $q \subseteq W_s, W_t \Rightarrow q \subseteq W_p$. То есть весь путь лежит в G_q. Так как вершины s и t выбирались произвольно, получим связность подграфа G_q. Из произвольности выбранной нагрузки q следует верность утверждения в целом, ч.т.д.

Следствие 1. *АПМГС строит алгоритм смежности.*

Доказательство.Заметим, перед началом работы цикла (8—22) структура данных Q содержит в себе Ω. Цикл (8—22) переберет все веса из Q. По первому утверждению получим, что по завершении цикла (8—22) $\forall q \in \Omega$ подграф G_q будет связным. Значит, построенный граф G будет являться графом смежности.

Остается показать минимальность построенного графа смежности.

Утверждение 3. *На каждой итерации по $q \in Q$ цикла (8—22) в граф G добавляются ребра с нагрузкой q.*

Доказательство. На каждой итерации по $q \in Q$ цикла (8—22) ребра могут быть добавлены только в подграф G_q, (см. (11—12)). Значит, для каждого добавляемого ребра e на текущей итерации $q \subseteq W_e$. Таким образом, из $|q| = |W_e|$ последует $q = W_e$.

Покажем, что $|q| = |W_e|$. Пусть мы хотим добавить ребро $e = (d, v)$ и $|W_e| > |q|$. Пусть алгоритм только выполнил строку (14). Ввиду порядка обработки весов граф G_{W_e} связен. Значит, вершины, инцидентные ребру e, находятся в одной компоненте связности. Но из условий, формируемых строками (12) и (14), следует, что $d \in S, v \notin S$. Из рассуждений, описанных выше, получим противоречие, так как S к моменту выполнения строки (14) является КС.

Значит, на каждой итерации цикла (8—22) для $q \in Q$ добавляются ребра веса q.

Утверждение 4. *АПМГС строит на множестве вершин V минимальный граф смежности.*

Доказательство. Для множества V рассмотрим некоторый произвольный граф смежности

$$G = \langle V, E \rangle.$$

АПМГС строит граф смежности

$$G^{\text{alg}} = \langle V, E^{\text{alg}} \rangle.$$

Каждому весу $q \in \Omega$ сопоставим множество ребер

$$\begin{aligned} S_q^{\text{alg}} &= \{e \in E^{\text{alg}} | W_e = q\}, \\ S_q &= \{e \in E | W_e = q\}. \end{aligned}$$

Примем во внимание, что системы попарно непересекающихся множеств $\{S_q^{\text{alg}}\}_{q \subseteq \Omega}$ и $\{S_q\}_{q \subseteq \Omega}$ суть разбиения множеств E^{alg} и E соответственно:

$$\begin{aligned} E^{\text{alg}} &= \cup_{q \in \Omega} S_q^{\text{alg}}, \\ E &= \cup_{q \in \Omega} S_q. \end{aligned}$$

Из свойств системы непересекающихся множеств следует:

$$\begin{aligned} |E^{\text{alg}}| &= \textstyle\sum_{q \in \Omega} |S_q^{\text{alg}}|, \\ |E| &= \textstyle\sum_{q \in \Omega} |S_q|. \end{aligned}$$

Таким образом, показав для каждого $q \in \Omega$ по отдельности, что $|S_q^{\text{alg}}| \leq |S_q|$, мы сможем сделать общий вывод: $|E^{\text{alg}}| \leq |E|$. Отсюда последует минимальность числа ребер в графе G^{alg}, а следовательно — корректность алгоритма.

Покажем, что для $\forall q \in \Omega\ |E_q^{\text{alg}}| \le |E_q|$. Рассмотрим два графа $G_q^{\text{alg}} = \langle V_q, E_q^{\text{alg}} \rangle$ и $G_q = \langle V_q, E_q \rangle$, индуцированные весом q над графами G^{alg} и G соответственно. Для них рассмотрим два подграфа

$$\begin{array}{lcl} H_{\text{alg}} & = & \langle V_q, E_q^{\text{alg}} \backslash S_q^{\text{alg}} \rangle, \\ H & = & \langle V_q, E_q \backslash S_q \rangle. \end{array}$$

Пусть АПМГС только приступил к итерации цикла (8—22) для текущей нагрузки q, но ребер в текущий граф G' еще не добавлял. Из третьего утверждения и порядка перебора элементов Ω последует, что $H_{\text{alg}} = G'_q$.

Рассмотрим вершины $s, t \in V_q$. Пусть в H существует путь P из s в t. Рассмотрим отдельно каждое ребро $e = (p_i, p_{i+1})$ данного пути с нагрузкой W_e. Далее рассмотрим подграф $G_{W_e}^{\text{alg}} = \langle V_{W_e}, E_{W_e}^{\text{alg}} \rangle$, индуцированный нагрузкой W_e. Ребро $e \in E_q \backslash S_q$, следовательно, $q \subset W_e$ строго. Значит, $V_{W_e} \subseteq V_q$ и $E_{W_e}^{\text{alg}} \subseteq E_q^{\text{alg}} \backslash S_q^{\text{alg}}$. Таким образом, $G_{W_e}^{\text{alg}}$ — подграф графа H_{alg}. Но граф $G_{W_e}^{\text{alg}}$ по первому утверждению связен. Значит, в графе H_{alg} между вершинами p_i и p_{i+1} есть путь. А значит, и между вершинами s и t существует путь в H_{alg}.

Из этого заключения можно сделать вывод: всякая КС графа H является подмножеством некоторой КС графа H_{alg}. То есть всего КС в H_{alg} не больше, чем в H.

Пусть число КС в H_{alg} будет k_{alg}, а в H — k. Каждый раз добавляя ребро, АПМГС соединяет две различные КС графа H_{alg}. Следовательно, всего ребер будет добавлено $k_{\text{alg}} - 1$. По третьему утверждению, $|S_q^{\text{alg}}| = k_{\text{alg}} - 1$. Мощность же $|S_q| \ge k - 1$ (так как G_q - связен). Но $k_{\text{alg}} \le k$. Значит, $|S_q^{\text{alg}}| \le |S_q|$. То есть $|E^{\text{alg}}| \le |E|$, следовательно, число ребер в графе G минимально, ч.т.д.

Если рассмотреть остовный лес построенного графа, то можно заметить: добавляя по ребру из G к дереву, мы увеличиваем число фундаментальных циклов в графе на единицу. А отсюда можно судить о минимальности числа циклов в графе G.

6. Заключение. В работе приведена спецификация алгоритма АПМГС, строящего минимальный граф смежности G по заданному множеству вершин и их весов. Основные шаги АПМГС состоят в следующем:

- построение универсального множества нагрузок Ω;

- перебор всех нагрузок q из полученного множества Ω в установленном порядке;

- модификация графа G по текущей нагрузке q. Модификация заключается в добавлении ребер в граф G с целью получить связный подграф G_q, индуцированный весом q.

Корректность АПМГС обоснована с помощью доказательства системы утверждений. Было установлено, что

- алгоритм выполняет модификации графа G по заданным нагрузкам корректно;

- граф является графом смежности тогда и только тогда, когда для всякой нагрузки $q \in \Omega$ подграф G_q связен;

- алгоритм строит граф смежности;

- проводя модификацию графа G по заданной нагрузке q, алгоритм добавляет ребра только с нагрузкой q.

Наконец, последнее утверждение показало, что АПМГС строит минимальный граф смежности, что служит обоснованием корректности алгоритма.

Контекст решаемой в статье задачи может накладывать дополнительные ограничения на получаемый граф смежности. Представляется возможным модифицировать полученный алгоритм с целью получения минимального графа смежности с минимальным диаметром.

Благодаря минимальности построенного графа можно судить о возможности формирования ациклической структуры алгебраической байесовской сети. Однако представляется возможным построить более быстрый алгоритм для определения данного свойства. Также представляется целесообразным поиск реализации алгоритма, оптимального по времени выполнения и занимаемой памяти.

Литература

1. *Павельчук А.В., Тулупьев А.Л., Тотмянина С.А.* Подход к объектно-ориентированному представлению данных алгебраических байесовских сетей в java-коде и реляционнных СУБД // Региональная

информатика–2008 (РИ-2008). XI Санкт-Петербургская международная конференция. Санкт-Петербург, 22–24 октября, 2008 г.: Материалы конференции / СПОИСУ. СПб., 2009. С. 68–76.

2. *Тотмянина С.А., Павельчук А.В., Тулупьев А.Л.* Алгебраические байесовские сети: структуры данных в СУБД и Java-коде // Интегрированные модели, мягкие вычисления, вероятностные системы и комплексы программ в искусственном интеллекте. Научно-практическая конференция студентов, аспирантов, молодых ученых и специалистов (Коломна, 26–27 мая 2009 г.). Научные доклады. В 2-х т. Т. 1. М.: Физматлит, 2009. С. 243–262.

3. *Тулупьев А.Л.* Дерево смежности с идеалами конъюнктов как ациклическая алгебраическая байесовская сеть // Труды СПИИРАН. Вып. 3, Т. 1. СПб.: Наука, 2006. С. 198–227.

4. *Тулупьев А.Л.* Ациклические алгебраические байесовские сети: логико-вероятностный вывод // Нечеткие системы и мягкие вычисления: Научный журнал Российской ассоциации нечетких систем и мягких вычислений. 2006. Том 1, № 1. С. 57–93.

5. *Тулупьев А.Л.* Алгебраические байесовские сети: глобальный логико-вероятностный вывод в деревьях смежности: Учеб. пособие. СПб.: СПбГУ; ООО Издательство «Анатолия», 2007. 40 с. (Сер. Элементы мягких вычислений).

6. *Тулупьев А.Л.* Байесовские сети: логико-вероятностный вывод в циклах. СПб.: Изд-во С.-Петербургского ун-та, 2008. 140 с. (Элементы мягких вычислений.)

7. *Тулупьев А.Л.* Задача локального автоматического обучения в алгебраических байесовских сетях: логико-вероятностный подход // Труды СПИИРАН. 2008. Вып. 7. СПб.: Наука, 2008. С. 11–25.

8. *Тулупьев А.Л.* Автоматическое обучение фрагментов знаний в алгебраических байесовских сетях // Интегрированные модели и мягкие вычисления в искусственном интеллекте. V-я Международная научно-практическая конференция. Сборник научных трудов. В 2-х т. Т. 1. С. 163–176.

9. *Тулупьев А.Л.* Матрично-векторные уравнения в задачах локального обучения алгебраических байесовских сетей // Региональная информатика–2008 (РИ–2008). XI Санкт-Петербургская международная конференция. Санкт-Петербург, 22–24 октября, 2008 г.: Материалы конференции / СПОИСУ. СПб., 2009. С. 91–99.

10. *Тулупьев А.Л.* Обработка дополнительной нечисловой информации в локальном обучении алгебраических байесовских сетей по выборкам с пропусками // Международная конференция по мягким вычислениям и измерениям. Сборник докладов. 2009. Т. 1. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2009. С. 139–142.

11. *Тулупьев А.Л.* Алгебраические байесовские сети: логико-вероятностная графическая модель баз фрагментов знаний с неопределенностью: Дисс. ... д-ра физ.-мат. наук по спец. 05.13.17 – Теор. осн. информатики. СПб., 2009. 670 с. (СПбГУ, мат.-мех. ф-т.)

12. *Тулупьев А.Л., Николенко С.И., Сироткин А.В.* Байесовские сети: логико-вероятностный подход. СПб.: Наука, 2006. 607 с.

13. *Тулупьев А.Л., Сироткин А.В.* Алгебраические байесовские сети: принцип декомпозиции и логико-вероятностный вывод в условиях неопределенности // Информационно-измерительные и управляющие системы. 2008. №10, Т. 6. С. 85–87.

14. *Тулупьев А.Л., Сироткин А.В., Николенко С.И.* Байесовские сети доверия: логико-вероятностный вывод в ациклических направленных графах. СПб.: Изд-во С.-Петерб. ун-та, 2009. 400 с.

15. *Тулупьев А.Л., Столяров Д.М., Ментюков М.В.* Представление локальной и глобальной структуры алгебраической байесовской сети в Java-приложениях // Труды СПИИРАН. 2007. Вып. 5. СПб.: Наука, 2007. С. 71–99.

16. *Фильченков А.А., Тулупьев А.Л.* Структурный анализ систем минимальных графов смежности // Труды СПИИРАН. 2009. Вып. 10. СПб.: Наука, 2009. С. 104–127.

17. *Korb K.B., Nicholson A.E.* Bayesian Artificial Intelligence. New York: Chapman and Hall/CRC, 2004. 364 p.

18. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦМНО. 1999. 960 с.

Опарин Всеволод Владиславович — студент факультета ИТиП СПбГУ ИТМО. oparin@rain.ifmo.ru; СПбГУ ИТМО, Кронверкский пр., д. 49, г. Санкт-Петербург, 19710, РФ; р.т. +7(812)328-3337, факс +7(812)328-4450. Научный руководитель — Тулупьев А.Л.

Oparin Vsevolod Vladislavovich — student of Informational Technologies and Programming Faculty of SPbSU ITMO. oparin@rain.ifmo.ru; SPbSU ITMO, Kronverkskiy pr., 49., St. Petersburg, 19710, Russia; office phone +7(812)328-3337, fax +7(812)328-4450. Scientific advisor — A.L. Tulupyev.

Тулупьев Александр Львович — к.ф.-м.н., доцент; и.о. заведующего лабораторией теоретических и междисциплинарных проблем информатики СПИИРАН, доцент кафедры информатики математико-механического факультета С.-Петербургского государственного университета (СПбГУ). Область научных интересов: представление и обработка данных и знаний с неопределенностью, применение методов математики и информатики в социокультурных исследованиях, применение методов биостатистики и математического моделирования в эпидемиологии, технология разработки программных комплексов с СУБД. Число научных публикаций — 160. ALT@iias.spb.su, www.tulupyev.spb.ru; СПИИРАН, 14-я линия В.О., д. 39, г. Санкт-Петербург, 199178, РФ; р.т. +7(812)328-3337, факс +7(812)328-4450.

Tulupyev Alexander Lvovich — PhD in Computer Science, Associate Professor; acting Head of Theoretical and Interdisciplinary Computer Science Laboratory, SPIIRAS, Associate Professor of Computer Science Department, SPbSU. Research area: uncertain data and knowledge representation and processing, mathematics and computer science applications in socio-cultural studies, biostatistics, simulation, and mathematical modeling applications in epidemiology, data intensive software systems development technology. Number of publications — 160. ALT@iias.spb.su, www.tulupyev.spb.ru; SPIIRAS, 14-th line V.O., 39, St. Petersburg, 199178, Russia; office phone +7(812)328-3337, fax +7(812)328-4450.

Поддержка исследований. Работа выполнена при финансовой поддержке РФФИ, проект № 09-01-00861-а.

Рекомендовано ЛПИ СПИИРАН, зав. лаб. Юсупов Р.М., член-корреспондент РАН.

Статья поступила в редакцию 12.01.2010.

РЕФЕРАТ

Оларин В.В., Тулупьев А.Л. **Синтез графа смежности с минимальным числом ребер: формализация алгоритма и анализ его корректности.**

В статье рассматриваются вопросы, связанные с одним из этапов машинного обучения алгебраических байесовских сетей (АБС). Рассматриваются проблемы в организации формальных моделей знаний с неопределенностью на примере алгебраических байесовских сетей (АБС).

Целью работы является построение вторичной структуры АБС в виде графа смежности. На граф смежности накладывается дополнительное ограничение: минимизовать число ребер. В статье приведен алгоритм построения минимального графа смежности, а также анализ его корректности.

В постановке задачи дается определение графа смежности, формализуются входные данные, ограничения и цель, вводится понятие веса вершины. Вводятся понятия нагрузки ребра, а также универсального множества нагрузок. Вводятся определение мощности веса, задается порядок на множестве весов вершин.

Приведен листинг алгоритма построения минимального графа смежности (АПМГС), описания используемых обозначений, а также построчный разбор АПМГС.

Сформулирован ряд утверждений, доказывающих корректность АПМГС. Доказывается, что АПМГС строит граф смежности. Затем доказывается минимальность построенного графа.

Установлено, что алгоритм по заданной первичной структуре АБС строит вторичную в виде графа смежности с минимальным числом ребер корректно.

SUMMARY

Oparin V.V., Tulupyev A.L. **Synthesis of a joint graph with minimal number of edges: an algorithm formalization and a proof of correctness.**

This paper relates to machine learning of algebraic Bayesian networks (ABN).

The paper's aim is to describe an algorithm for synthesis of the secondary structure of algebraic Bayesian network as a joint graph. There is an additional constraint: the joint graph must have minimal number of edges. After the algorithm description, it's analysis is done.

Next, we formalize input data, problem statement, and goal. Then we introduce definitions of joint graphs and weight of vertex, and later on definitions of specific elements of graph theory. Load, universal set of loads, and also the order of weights are introduced.

The last section gives a proof of the algorithm correctness. The first two statements prove that the algorithm builds a joint graph. The next two ones prove that the synthesized joint graph contains minimal number of edges.