УДК 004.8

А.А. ФИЛЬЧЕНКОВ

АЛГОРИТМЫ ПОСТРОЕНИЯ ТРЕТИЧНОЙ СТРУКТУРЫ АЛГЕБРАИЧЕСКОЙ БАЙЕСОВСКОЙ СЕТИ

Фильченков А.А. **Алгоритмы построения третичной структуры алгебраической байесовской сети.**

Аннотация. Третичная структура алгебраической байесовской сети (АБС), представляемая в виде графа клик, важна для построения и анализа вторичной структуры АБС, а также для анализа ее первичной структуры. В статье предложены два алгоритма построения третичной структуры: *алгоритм построения графа клик при помощи потомков* и *алгоритм построения графа клик снизу—вверх*, доказана их корректность и оценено время работы. Оба алгоритма по заданному набору максимальных фрагментов знаний строят два упорядоченных множества, содержащие множества вершин и множества сыновей каждой клики. Приведены примеры первичных структур АБС, на которых первый алгоритм работает быстрее второго и, наоборот, второй — быстрее первого. Также установлены существование и единственность третичной структуры АБС для каждой первичной структуры АБС.

Ключевые слова: алгебраические байесовские сети, третичная структура, машинное обучение, вероятностно-графические модели систем знаний, глобальная структура.

Filchenkov A.A. **Algebraic Bayesian Network Tertiary Structure Synthesis Algorithms.**

Abstract. Algebraic Bayesian network (ABN) tertiary structure represented as a clique graph is important for synthesizing and analyzing the ABN secondary structure, as well as for an analysis of the ABN primary structure. Two tertiary structure synthesis algorithms are proposed: clique tree synthesis descendants algorithm, and clique tree synthesis bottom-up algorithm, their validity is proved and their computational complicity is estimated in the article. Both the algorithms synthesize two ordered sets containing the set of vertices and the set of sons of each clique for given maximal knowledge pattern set Examples of the ABN primary structures for which the first algorithm works faster than the second one, and vice versa, the second one works faster than the first one, are given. Also ABN tertiary structure existence and uniqueness for every ABN primary structure are stated.

Keywords: algebraic Bayesian networks, tertiary structure, machine learning, probabilistic graphical models, global structure.

1. Введение. Алгебраические байесовские сети (АБС) представляют собой вероятную графическую модель баз фрагментов знаний с неопределенностью [11, 16, 18, 37]. АБС позволяют осуществлять различные виды логико-вероятностного вывода [6–8, 11] и поддерживать непротиворечивость (локальную, интернальную, экстернальную, глобальную) на наборе фрагментов знаний [13, 14, 17, 19]. Основным отличием АБС от других вероятностных графических моделей является возможность работы с интервальным оценками вероятностей [5, 9, 12, 15].

АБС может моделировать работу родственных ей вероятностных графических моделей, в частности, скрытых марковских моделей [2, 3], которые нашли широкое применение в области распознавания речи [34, 35]. Другой пример возможного применения — распознавание механизмов распространения сетевых червей, для которого в случае детерминированных процессов используются сети Петри [36], тогда как для случая недетерминированных процессов, когда возникает неопределенность, может потребоваться использование АБС.

Первичной структурой АБС называют набор фрагментов знаний, над которым строится сеть, а вторичной — граф над первичной структурой, который также называют *графом смежности* [4, 20, 21, 29]. Одной из задач глобального обучения является построение вторичной структуры над заданной первичной структурой, и некоторые из путей решения этой задачи [24, 25, 27, 30] требуют построения третичной структуры — направленного графа, вершины которого соответствуют полным подграфам максимального графа смежности, построенного над данной первичной структурой [10, 21, 26]. Такой граф также называется графом клик.

Построение графа клик может осуществляться либо как составная часть построения вторичной структуры, либо отдельно от него для оценки качества первичной структуры, где под качеством понимается отсутствие циклов в какой-либо вторичной структуре, построенной над данной первичной[1] [26].

Цель данной работы — разработать алгоритмы построения графа клик и описать их свойства.

2. Основные определения, обозначения. Мы будем следовать системе терминов, введенной в [28, 31–33].

Как и сама АБС, третичная структура АБС строится над *набором главных конъюнктов максимальных фрагментов знаний (набор МФЗ)* представляющих собой такое множество слов, построенных над заданным алфавитом, что никакое слово полностью не содержит никакое другое слово. МФЗ также являются вершинами графа, образованного вторичной структурой, и далее мы будем использовать слово вершина именно в этом понимании, если не оговорено обратное.

[1] Циклы во вторичной структуре не позволяют осуществить некоторые виды логико-вероятностного вывода, поэтому являются серьезной преградой для работы АБС. Именно поэтому те первичные структуры, над которыми невозможно построение ациклической вторичной структуры, являются неудачной основой для создания глобальной структуры АБС.

Вес $W(V)$ вершины V — множество атомов алфавита, вошедших в соответствующий МФЗ. По сути, вес вершины совпадает с самим МФЗ, данное определение вводится для удобства. Над множеством весов определены те же самые операции, что и над множествами: объединение, пересечение, включение и т. д. Будем говорить, что вес u_1 *содержится* в весе u_2, если $u_1 \subseteq u_2$.

Помимо этого, над заданным набором МФЗ можно ввести индексирующую функцию I, которая каждому весу будет сопоставлять натуральное число, представляемой в двоичной записи соответствующим весом [21, 28]. Так, например, набор МФЗ $\{ax, ay, xz\}$, заданный, соответственно, над алфавитом $\{a, x, y, z\}$, индексируется следующим образом:

$$I(ax) = 1100_2 = 12;$$
$$I(ay) = 1010_2 = 10;$$
$$I(yz) = 0011_2 = 3.$$

Вес, равный пересечению весов двух вершин, будем называть *значимым весом*.

Клика веса U — полный граф, построенный на максимальном по включению множестве вершин, веса которых содержат значимый вес U (рис. 1). Множество всех клик будем обозначать как Clique. Вес клики P будем обозначать как $W(Q)$. Будем говорить, что клика P *содержит* клику Q, если $W(Q)$ содержит $W(P)$ или, что то же самое, множество вершин клики P содержит множество вершин клики Q. Следует отметить, что любой значимый вес будет являться весом какой-либо клики.

Если клика P содержит клику Q, то клика P называется *предком* клики Q, а клика Q — *потомком* клики P. Важно обратить внимание на то, что индекс веса предка меньше индекса веса потомка.

Если клика P является предком клики Q, и не существует такой клики R, что R является одновременно потомком P и предком Q, то клика P называется *родителем* клики Q, а клика Q — *сыном* клики P.

Граф клик — направленный граф, вершинами которого являются клики из множества Clique (рис. 1). Ребро из вершины P в вершину Q проведено, если клика P является родителем клики Q.

Мы будем считать, что сыновья клики упорядочены по индексу их веса.

Утверждение 1. Для каждой первичной структуры граф клик существует и единственен.

Доказательство. Заметим, что множество Clique определено однозначно, поскольку содержит все элементы (которых может и не

быть), удовлетворяющие определенному условию. Для каждой первичной структуры можно построить, и при том единственное, множество Cliuqe.

Докажем, что над каждым множеством Clique можно построить граф с заданным в определении графа клик условием. Прежде всего заметим, что для произвольных весов u и w не может одновременно выполняться $u \subset w$ и $w \subset u$ (так как это конечные множества), и кроме того, ни для каких двух различных u и w весов $u \subseteq w$ (это следует из условия максимальности фрагментов знаний [28]). Следовательно, для любых двух клик мы можем однозначно указать, является ли первая потомком второй либо вторая — потомком первой, либо они не состоят в подобных отношениях. Далее, докажем, что для каждой клики множество ее сыновей задается однозначно и непротиворечиво. Поскольку отношения предок–потомок является частичным порядком, то на множестве потомков клики можно указать подмножество минимальных элементов, которое единственно, и которое как раз и является множеством сыновей данной клики.

Таким образом, мы доказали, что граф клик над данной первичной структурой существует и единственен.

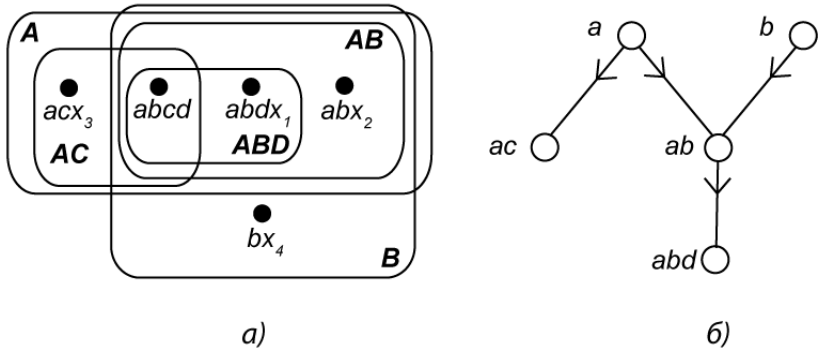

Рис. 1. Клики и граф клик.
На рис. 1.*а* изображены клики, построенные над вершинами $abcd$, $abdx_1$, abx_2, acx_3, bx_4. На рис. 1.*б* изображен соответствующий граф клик.

Основное множество вершин клики с весом u — множество таких вершин, что для каждой вершины b этого множества существует отличная от нее вершина c, что $W(b) \cap W(c) = u$.

Замечание 1. Все вершины из основного множества вершин клики принадлежат ей.

Например, для клики веса a, состоящей из трех вершин abc, abx, acy, вершина abc имеет пересечение с другими вершинами по весам ab и ac соответственно, поэтому не входит в основное множество вершин клики с весом a, тогда как остальные две вершины пересекаются по весу a, поэтому входят в это множество.

Утверждение 2. Каждая вершина, входящая в клику веса u, в ходит в множество основных вершин либо самой клики, либо в множество основных вершин какого-либо ее потомка.

Доказательство. Поскольку u — значимый вес, то можно указать две вершины, пересечение весов которых в точности равно u. Назовем одну из этих вершин a. Рассмотрим произвольную вершину b, содержащую вес u. Она будет содержаться в клике веса $W(a) \cap W(b)$, и, более того, будет принадлежать основному множеству вершин этой клики. Поскольку $u \subset W(a)$ и $u \subset W(b)$, то $u \subset W(a) \cap W(b)$, следовательно, клика веса $W(a) \cap W(b)$ является потомком клики веса u.

Будем считать, что на вход алгоритмов поступает неупорядоченное множество МЗФ Weights, элементы которого представляются битовыми последовательностями фиксированной длины. Последнее гарантирует нам, что объединение и пересечение двух весов, а также проверка включения одного веса в другой, выполняются за $O(1)$ (t_U, t_I, t_C соответственно).

Требуемый граф клик будем представлять в виде двух упорядоченных по индексу веса множеств Vertecies и Sons, где Vertecies$[u]$ содержит вершины, принадлежащие клике с весом u, а Sons$[u]$ — сыновей той же клики.

Дополнительно будем считать, что все множества, создающиеся по ходу выполнения алгоритма, являются упорядоченными, и любая операция (объединение, пересечение, добавление, поиск) производится над упорядоченным множеством. Множества Vertecies, Sons и Descendants могут быть организованы, например, хэш-таблицей [1], предоставляя доступ к своим элементам по ключам, соответствующим индексам веса этих элементов (вместо Vertecies$[I(u)]$ мы будем писать просто Vertecies$[u]$). Все остальные встречающиеся множества просто упорядочены по весу их элементов, но при этом организованы в линейный массив с диапазоном индексов $1 \ldots n$, где n — число элементов, содержащихся в массиве.

3. Алгоритм построения графа клик при помощи потомков. *Алгоритм построения графа клик при помощи потомков (clique tree synthesis descendants algorithm)* строит по заданному набору МФЗ Weights, граф клик, представляемый множеством множеств вершин каждой клики и множеством множеств сыновей каждой клики Vertecies и Sons соответственно.

```
Require: Weights
Ensure: ∄w1, w2 ∈ Weights: w1 ⊆ w2
 1: UsefulWeights ← ∅
 2: for all w1, w2 ∈ Weights, w1 ≠ w2 do
 3:    w ← w1 ∩ w2
 4:    if w ≠ ∅ do
 5:       if w ∉ UsefulWeights do
 6:          UsefulWeights ← UsefulWeights ∪ {w}
 7:          Vertecies[w] ← ∅
 8:          Descendants[w] ← ∅
 9:       end if
10:       Vertecies[w] ← Vertecies[w] ∪ {w1}
11:       Vertecies[w] ← Vertecies[w] ∪ {w2}
12:    end if
13: end for
14: for all u1, u2 ∈ UsefulWeights, u2 ⊄ u1 do
15:    if u1 ⊂ u2 do
16:       Vertecies[u1] ← Vertecies[u1] ∪ Vertecies[u2]
17:       Descendants[u1] ← Descendants[u1] ∪ {u2}
18:    end if
19: end for
20: for all u ∈ UsefulWeights do
21:    PraDescendants ← ∅
22:    for all w ∈ Descendants[u] do
23:       PraDescendants ← PraDescendants ∪ Descendants[w]
24:    end for
25:    Sons[u] ← Descendants[u] ∩ PraDescendants
26: end for
27: return Vertecies, Sons
```

Листинг 1. Алгоритм построения графа клик при помощи потомков.

В цикле (2–13) строится множество значимых весов UsefulWeights, а также для каждого значимого веса u строится множе-

ство Vertecies$[u]$, которое после окончания цикла будет содержать основное множество вершин клики веса u.

Условный оператор (4–12) в случае, если пересечение весов двух вершин непусто (тогда это пересечение является значимым весом), добавляет пару этих вершин к соответствующей клике.

Условный оператор (5–9) в случае, если значимый вес u еще не содержится во множестве значимых весов UsefulWeights, добавляет его туда и создает пустые множества вершин Vertecies$[u]$ и потомков Descendants$[u]$ в ячейках, соответствующих весу u.

В цикле (14–19) для каждой клики u строится множество ее потомков Descendants$[u]$, а также окончательно заполняется множество ее вершин путем добавления к Vertecies$[u]$, сформированному на предыдущих шагах, множеств вершин всех потомков этой клики.

В цикле (20–26) для каждой клики u строится множество ее сыновей Sons$[u]$ путем исключения из Descendants$[u]$ всех ее потомков, которые являются сыновьями каких-либо других ее потомков (множество PraDescendants).

В цикле (22–24) строится упомянутое множество потомков клики, являющихся сыновьями каких-либо других ее потомков.

Утверждение 3. Алгоритм построения множества графа клик при помощи потомков строит граф клик.

Доказательство. Поскольку алгоритм конечен, достаточно доказать, что для каждого значимого веса u множества Vertecies$[u]$ и Sons$[u]$ будут содержать соответственно принадлежащие клике вершины и ее сыновей и только их.

Сначала докажем, что в UsefulWeights будут содержаться значимые веса и только они. По построению в это множество попадают те и только те веса, которые представляются пересечением весов каких-либо двух вершин, что совпадает с определением значимого веса.

Теперь докажем, что в Sons$[u]$ попадут веса сыновей клики u и только они. В цикле (14–19) в Descendants$[u]$ попадут все такие и только такие веса w, для которых $u \subset w$, что совпадает с определением потомка.

В цикле (22–24) множество PraDescendnts будет сформировано из тех и только тех потомков клики веса u, которые при этом являются потомками какого-либо потомка клики веса u. Иначе говоря, для каждого потомка Q, который попадает в это множество, существует клика R, которая является предком Q и потомком клики веса u. Все потомки клики u, не попавшие в это множество, являются по определению ее

сыновьями. Поэтому на шаге (25) строится множество сыновей клики веса u.

Теперь докажем, что $\mathrm{Vertecies}[u]$ будет состоять только из вершин, вес которых содержит u. После завершения цикла (2–13) в $\mathrm{Vertecies}[u]$ будут содержаться все вершины из основного множества вершин клики веса u, поскольку они добавляются парами, пересечение весов которых в точности равно u. После завершения цикла (14–18) к $\mathrm{Vertecies}[u]$ будут добавлены вершины всех потомков клики веса u, т. е. согласно утверждению 2, все оставшиеся вершины клики веса u, не входящие в ее основное множество вершин. Отсюда следует, что $\mathrm{Vertecies}[u]$ будет содержать все вершины, вес которых содержит u. Так как веса всех вершин, которые попадают в $\mathrm{Vertecies}[u]$ после завершения цикла (2–13) содержат вес u, а после завершения цикла (14–18) к ним добавятся вершины, содержащие вес их потомков (и, следовательно, их самих), то $\mathrm{Vertecies}[u]$ будет содержать только вершины веса u.

Утверждение 4. Алгоритм работает за не более чем

$$2|W|^2 + 2|U|\log_2|U| + 2|U|^2 + \\ + \sum_{u \in U}[2e_U(\log_2|U| + \log_2|V_u|) + |D_u|(2\log_2|D_u| + |A_u|) + \\ + |V_u|(1 + |A_u|)],$$

где e_u — число пар вершин, вес которых равен u U — множество $\mathrm{UsefulWeights}$; V_u — множество вершин клики с весом u; D_u — множество потомков клики с весом u, A_u — множество предков клики с весом u.

Доказательство. На выполнение шагов (3) и (4) потребуется в сумме

$$2|W|^2 \tag{1}$$

поскольку две операции будут осуществляться для каждой пары вершин.

На выполнение проверки на шаге (5) потребуется

$$2\log_2|U| \cdot \sum\nolimits_{u \in U} e_U, \tag{2}$$

т. к. она будет осуществляться ровно $\sum_{u \in U} e_U$ раз (для всех пар, пересечение весов которых непусто, т. е. образует значимый вес), требуя $2\log_2|U|$ операций для поиска элемента в упорядоченном множестве U.

На формирование упорядоченного множества U на шаге 6 потребуется

$$2|U| \cdot \log_2|U|. \tag{3}$$

На выполнение шагов (10–11) потребуется

$$\sum\nolimits_{u \in U} (2 \log_2|V_u| \cdot e_u), \tag{4}$$

поскольку для каждого значимого веса u в множество V_u будет осуществлено e_u добавлений вершин.

На выполнение проверки на шаге (14) потребуется

$$|U|^2, \tag{5}$$

поскольку для каждой пары значимых весов необходимо провести эту проверку, тогда как на шаге (15) лишь

$$|U| \cdot \left(|U| - \sum\nolimits_{u \in U} |D_u|\right), \tag{6}$$

потому что она для каждой клики веса u осуществляется только для тех клик, которые не являются ее предками, а сумма числа потомков всех клик равна числу предков всех клик.

На выполнение шага (16) потребуется не более

$$\sum\nolimits_{u \in U} \left(|V_u| + \sum\nolimits_{d \in D_u} |V_d|\right), \tag{7}$$

потому что для каждого значимого веса u мы к упорядоченному множеству вершин, в котором не более чем $|V_u|$ элементов, будем добавлять упорядоченные множества вершин его сыновей d, в которых не более чем $|V_d|$ элементов.

На выполнение шага 17 потребуется

$$\sum\nolimits_{u \in U} (2|D_u| \cdot \log_2|D_u|), \tag{8}$$

потому что для каждого значимого веса u формируется упорядоченное множество его потомков.

На выполнение шага 23 потребуется не более

$$\sum_{u \in U} \sum_{d \in D_u} |D_d|, \tag{9}$$

поскольку для каждого значимого веса u PraDescendants будет получено слиянием упорядоченных множеств потомков его потомков.

На выполнение шага 25 потребуется не более

$$\sum_{u \in U} \left(|D_u| + \sum_{d \in D_u} |D_d| \right), \tag{10}$$

потому что для каждого значимого веса u упорядоченное множество D_u пересекается с упорядоченным множеством PraDescendants, в котором не более, чем $\sum_{d \in D_u} |D_d|$ элементов.

Теперь осталось только заметить, что в формулах (7), (9) и (10) суммирование $\sum_{u \in U} \sum_{d \in D_u} x_d$ можно свернуть до $\sum_{u \in U} A_u x_u$, потому что, как уже было отмечено, число сумм по предкам всех клик совпадает с суммой по потомкам всех клик.

Сложив и сгруппировав формулы (1)–(10), мы получим искомое выражение.

4. Алгоритм построения графа клик снизу—вверх. *Алгоритм построения графа клик снизу—вверх (clique tree synthesis bottom-up algorithm)* строит по заданному набору МФЗ Weights, граф клик, представляемый множеством множеств вершин каждой клики и множеством множеств сыновей каждой клики Vertecies и Sons соответственно.

Require: Weights
Ensure: $\nexists w_1, w_2 \in \text{Weights}: w_1 \subseteq w_2$
1: UsefulWeights $\leftarrow \emptyset$
2: **for all** $w_1, w_2 \in \text{Weights}, w_1 \neq w_2$ **do**
3: $w \leftarrow w_1 \cap w_2$
4: **if** $w \neq \emptyset$ **do**
5: **if** $w \notin$ UsefulWeights **do**
6: UsefulWeights $\leftarrow$ UsefulWeights $\cup \{w\}$
7: Vertecies$[w] \leftarrow \emptyset$
8: Sons$[w] \leftarrow \emptyset$
9: **end if**
10: Vertecies$[w] \leftarrow$ Vertecies$[w] \cup \{w_1\}$
11: Vertecies$[w] \leftarrow$ Vertecies$[w] \cup \{w_2\}$
12: **end if**
13: **end for**
14: **for** $i \leftarrow |\text{UsefulWeights}|$ **downto** 1 **do**
15: $u \leftarrow$ UsefulWeigths$[i]$
16: **for** $j \leftarrow i + 1$ **to** $|\text{UsefulWeights}|$ **do**
17: $w \leftarrow$ UsefulWeights$[j]$
18: **if** $u \subset w$ **do**
19: $is\_grson \leftarrow$ FALSE

```
20:         for all s ∈ Sons[u] do
21:           if s ⊂ w do
22:             is_grson ← TRUE
23:             break
24:           end if
25:         end for
26:         if is_grson = FALSE do
27:           Vertecies[u] ← Vertecies[u] ∪ Vertecies[w]
28:           Sons[u] ← Sons[u] ∪ {w}
29:         end if
30:       end if
31:     end for
32: end for
33: return Vertecies, Sons
```

Листинг 2. Алгоритм построения графа клик снизу—вверх.

Цикл (2–13) не отличается от такого же цикла в предыдущем алгоритме, в нем строится множество значимых весов UsefulWeight и для каждой клики веса u — множество $\mathrm{Vertecies}[u]$, которое после завершения цикла будет совпадать с основным множеством вершин этой клики.

В цикле (14–32) последовательно перебираются значимые веса по убыванию индекса (т. е. от листьев дерева клик к его вершинам) и для каждого значимого веса u формируется соответствующее ему множество его сыновей $\mathrm{Sons}[u]$, а также достраивается множество вершин $\mathrm{Vertecies}[u]$.

В цикле (16–31) перебираются все значимые веса w, индекс которых больше индекса веса u, выбранного на очередном шаге цикла (14–23), и на основе всех клик с такими весами w, являющихся сыновьями клики с весом u, строятся множество вершин и множество сыновей последней.

Условный оператор (18–30) в случае, если вес u содержится в w, проводит проверку на то, является ли он потомком какого-либо сына (за это отвечает переменная is_grson) и обрабатывает его, если он таковым не является.

В цикле (20–25) перебираются элементы множества $\mathrm{Sons}[u]$, и если один из них оказавается предком веса w, что проверяет условный оператор (21–24) то переменной is_grson присваивается истина, а цикл прерывается.

Условный оператор (26–29) в случае, если не нашлось сына из $\mathrm{Sons}[u]$, который бы являлся потомком клики веса w, добавляет ее к множеству сыновей клики веса u $\mathrm{Sons}[u]$, а все вершины клики веса w — в множество вершин $\mathrm{Vertecies}[u]$.

Утверждение 5. Алгоритм построения графа клик снизу—вверх строит граф клик.

Доказательство. Как известно из доказательства утверждения 3, после завершения цикла (2–13) будет построено множество значимых весов UsefulWeights и для каждой клики веса u множество $\mathrm{Vertecies}[u]$ будет совпадать с основным множеством вершин этой клики.

Теперь докажем, что $\mathrm{Sons}[u]$ будет содержать всех сыновей клики с весом u и только их. В цикле (16–31), при построении $\mathrm{Sons}[u]$ рассматриваются только те клики, веса которых по индексу больше са u, т. к. вес любого сына клики больше ее собственного веса. По этой же причине для каждого потомка, не являющегося сыном клики с весом u, вес его предка, являющегося сыном клики с весом u, будет обработан в этом цикле раньше, поэтому если в $\mathrm{Sons}[u]$ попадут все сыновья клики веса u, то в ней окажутся только они.

Докажем, что в $\mathrm{Sons}[u]$ попадут все сыновья соответствующей клики. Так как ни один из сыновей не является потомком другого сына (или даже какого-либо потомка клики), то после выполнения цикла (20–25) значение переменной is_grson окажется ложью, поэтому сын добавится к множеству $\mathrm{Sons}[u]$.

Теперь докажем, что $\mathrm{Vertecies}[u]$ после завершения цикла (16–31) будет содержать все вершины клики с весом u и только их. Как уже было сказано, к началу этого цикла $\mathrm{Vertecies}[u]$ совпадает с основным множеством вершин клики веса u. Поскольку в цикле (14–32) перебор осуществляется по убыванию индексов, то к моменту рассмотрения клики веса u множество $\mathrm{Vertecies}[s_i]$ будет сформировано для всех ее сыновей s_i. Проведем индукцию, доказывая, что $\mathrm{Vertecies}[u]$ будет содержать только вершины клики веса u и ничего, кроме них. Базой индукции будут бездетные клики, то есть клики, у которых нет сыновей, и, следовательно, по утверждению 2, множество вершины совпадает с основным множеством вершин. Для них доказываемое утверждение верно в силу того, что $\mathrm{Vertecies}[u]$ после завершения цикла (2–13) не будет меняться и, следовательно, будет содержать только основное множество вершин, которое, как уже сказано, совпадает с множеством всех вершин таких клик. Теперь осуществим переход, предполагая, что для всех сыновей s_i клики u множества $\mathrm{Vertecies}[s_i]$ сов-

падают с множеством их вершин. Любая вершина, не принадлежащая основному множеству вершин, содержащаяся в одном из потомков клики, будет также содержаться в одном из ее сыновей, потому как все вершины, принадлежащие потомкам клики, принадлежат множеству ее сыновей (это, в частности, следует из теоремы о классификации владений [22]). Благодаря этому на шаге (27) все вершины, не входящие в основное множество вершин клики веса u, добавятся к множеству $\mathrm{Vertecies}[u]$.

Мы доказали, что для любой клики веса u множества $\mathrm{Sons}[u]$ и $\mathrm{Vertecies}[u]$ будут совпадать с множествами вершин и сыновей клики веса u.

Утверждение 6. Алгоритм работает за не более чем

$$2|W|^2 + 2|U|\log_2|U| + \frac{|U|^2}{2} + \\ + \sum_{u \in U}[2e_u(\log_2|V_u| + \log_2|U|) + |V_u|(1 + |P_u|) + \\ + |S_u|(|D_u| + 2\log_2|S_u|)],$$

где e_u — число пар вершин, вес которых равен u U — множество $\mathrm{UsefulWeights}$; V_u — множество вершин клики с весом u; D_u — множество потомков клики с весом u, S_u — множество сыновей клики с весом u, P_u — множество родителей клики с весом u.

Доказательство. Как следует из доказательства утверждения 4, на выполнение шагов (2–13) потребуется не более

$$2|W|^2 + 2|U|\log_2|U| + \sum_{u \in U}(2e_u(\log_2|V_u| + \log_2|U|)). \quad (11)$$

На выполнение проверки на шаге (18) потребуется

$$\frac{|U|^2}{2}, \quad (12)$$

так как она будет осуществлена для всех пар (i, j), таких, что $i < j$, где i, j — индексы элементов множества U.

В худшем случае на выполнение всех циклов (20–25) потребуется

$$\sum\nolimits_{u \in U} |D_u| \cdot |S_u|, \quad (13)$$

так как для клики веса u будет требоваться осуществить проверку на шаге (21) на принадлежность каждого ее потомка каждому ее сыну.

На выполнение шага (27) потребуется

$$\sum_{u\in U}\left(|V_u| + \sum_{s\in S_u}|V_s|\right), \tag{14}$$

потому что для каждой клики веса u строится упорядоченное множество ее вершин добавлением к упорядоченному основному множеству ее вершин (мощность которого не больше, чем $|V_u|$) упорядоченных основные множеств вершин ее сыновей.

На выполнение шага (28) потребуется

$$\sum_{u\in U}(2|S_u|\log_2|S_u|), \tag{15}$$

поскольку для каждой клики веса u строится множество ее сыновей поэлементным добавлением.

Заменив из тех же соображений, каким мы руководствовались в доказательстве утверждения 4, $\sum_{u\in U}\sum_{s\in S_u}|V_s|$ на $\sum_{u\in U}|V_u||P_u|$, сложив формулы (11)–(15) и сгруппировав результат, получим искомую формулу.

5. Сравнение времени работы алгоритмов. Попробуем сравнить время работы алгоритмов. Для этого вычтем скорость работы второго из скорости работы первого:

$$2|U|^2 + \sum_{u\in U}[|D_u|(2\log_2|D_u| + |A_u|) + |V_u|\cdot|A_u|] -$$
$$-\frac{|U|^2}{2} + \sum_{u\in U}[\,|S_u|(2\log_2|S_u| + |D_u|) + |V_u|\cdot|P_u|],$$

и разобьем это по слагаемым:

$$\frac{3|U|^2}{2} +$$
$$+\sum_{u\in U}(2|D_u|\log_2|D_u| - 2\,|S_u|\log_2|S_u|) +$$
$$+\sum_{u\in U}|V_u|(|A_u| - |P_u|) +$$
$$+\sum_{u\in U}|D_u|(|A_u| - |S_u|).$$

Очевидно, что $\frac{3|U|^2}{2} > 0$. $\sum_{u\in U}|V_u|(|A_u| - |P_u|) \geq 0$, так как $|A_u| \geq |P_u|$, поскольку для любой клики любой ее родитель является ее потомком. $\sum_{u\in U}(2|D_u|\log_2|D_u| - 2\,|S_u|\log_2|S_u|) \geq 0$, так как $|D_u| \geq |S_u|$, потому что для любой клики любой ее сын является ее потомком.

Однако последнее слагаемое $\sum_{u \in U} |D_u|(|A_u| - |S_u|)$ вовсе необязательно больше нуля, и мы приведем пример, когда второй алгоритм работает медленнее.

Рассмотрим граф клик, состоящий из $2n$ клик, в котором n вершин являются отцами каждой из оставшихся n вершин (т.е. полный двудольный граф) (рис. 2).

Для такого графа для каждой клики веса u верно $|A_u| = |P_u|$ и $|D_u| = |S_u|$, для клик–«отцов» $|A_u| = 0$ и $|D_u| = n$, для клик–«детей» наоборот: $|A_u| = n$ и $|D_u| = 0$, наконец, $|U| = 2n$.

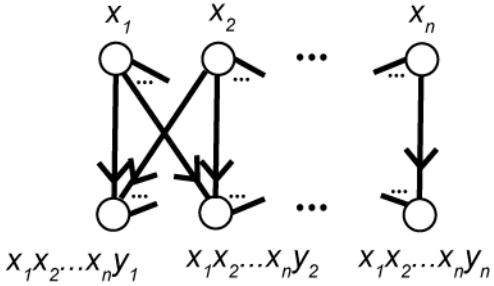

Рис. 2. Граф клик из $2n$ вершин, высота которого равна 1.

Итак, второе и третье слагаемые обнулятся, первое слагаемое станет равно $6n^2$, а в четвертом слагаемом в сумме слагаемые «отцов» будут равны $-n^2$, а слагаемые «сыновей» — 0. Таким образом, четвертое слагаемое будет равно $-n^3$, а общая сумма будет равна $6n^2 - n^3$, что меньше нуля при $n > 6$.

Теперь приведем пример, на котором второй алгоритм работает быстрее. Те же самые 2n вершин мы выстроим в одну цель длиной 2n, то есть когда каждая клика кроме первой и последней является сыном ровно одной клики и отцом ровно одной клики, а первая — отцом ровно одной клики (рис. 3). Для i–й клики в этой последовательности $|D_u|(|A_u| - |S_u|)$ будет равно $(2n - i) \cdot (i - 1 - 1) = (2n - i)(i - 2)$. $\sum_{i=1}^{2n}(2n - i)(i - 2) > 0$ уже при $n > 2$.

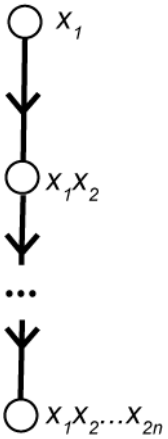

Рис. 3. Граф клик из $2n$ вершин, высота которого равна $2n - 1$.

Для сравнения алгоритмов требуется провести анализ структуры графа клик, выяснив отношения зависимости между такими значениями, как число сыновей, число потомков, число родителей, число предков, число вершин для каждой клики.

Уже сейчас можно говорить о том, что первый алгоритм быстрее второго на графов клик в первую очередь с маленькой высотой [2] и большим количеством ребер между вершинами. В случае большой высоты графа клики или даже просто небольшого числа ребер преимущество в скорости работы будет у второго алгоритма.

6. Заключение. В статье были рассмотрены два алгоритма построения графа клики и приведена оценка сложности их работы, которая была выражена через ряд параметров: число ребер определенного веса, число вершин клики, число сыновей клики, число родителей клики, число потомков клики и число предков клики, число значимых весов и, наконец, число МФЗ в наборе.

Обобщающие оценки сложности работы алгоритмов возможно получить, только осуществив анализ третичной структуры и выявив соотношения между указанными параметрами. Подобный анализ в отношении внутренней структуры клик [29, 32] позволил сформулиро-

[2] Под высотой графа клик естественным образом понимается высота соответствующего направленного графа, то есть длина самого длинного направленного пути в графе.

вать новые подходы к построению множества графов смежности [22–25].

Подобный анализ нужен также с точки зрения сравнения времени работы алгоритмов и точного указания тех случаев, когда один алгоритм работает эффективнее другого.

Улучшение времени работы алгоритма построения графа клик, которое позволит сократить время на построение и анализ третичной структуры, также является желательным с точки зрения развития теории АБС.

Также в статье были установлены существование и единственность третичной структуры АБС для любой первичной структуры АБС. Над первичной структурой АБС возможно построение множества вторичных структур [21, 27], с точки зрения чего третичная структура АБС инварианта ее первичной структуре.

Литература

1. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы. Построение и анализ. 2-е изд. М.: Вильямс. 2005. 1296 с.
2. *Момзикова М.П., Великодная О.И., Пинский М.Я., Сироткин А.В., Тулупьев А.Л., Фильченков А.А.* Оценка вероятности наблюдаемой последовательности в бинарных линейных по структуре скрытых марковских моделях с помощью апостериорного вывода в алгебраических байесовских сетях // Труды СПИИРАН. 2010. Вып. 2 (13). С. 122–142.
3. *Момзикова М.П., Великодная О.И., Пинский М.Я., Сироткин А.В., Тулупьев А.Л., Фильченков А.А.* Представление бинарных линейных по структуре скрытых марковских моделей в виде алгебраических байесовских сетей // Труды СПИИРАН. 2010. Вып. 1 (12). С. 134–150.
4. *Опарин В.В., Фильченков А.А., Тулупьев А.Л., Сироткин А.В.* Матроидное представление семейства графов смежности над набором фрагментов знаний // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2010. Вып. 4. С. 73–76.
5. *Сироткин А.В.* Модели, алгоритмы и вычислительная сложность синтеза согласованных оценок истинности в алгебраических байесовских сетях // Информационно-измерительные и управляющие системы. 2009. №11. С. 32–37.
6. *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).
7. *Тулупьев А.Л.* Алгебраические байесовские сети: система операций локального логико-вероятностного вывода // Информационно-измерительные и управляющие системы. 2009. №4. С. 41–44.
8. *Тулупьев А.Л.* Алгебраические байесовские сети: система операций глобального логико-вероятностного вывода // Информационно-измерительные и управляющие системы. 2010. №11. С. 65–72.
9. *Тулупьев А.Л.* Апостериорные оценки вероятностей в идеале конъюнктов // Вестник СПбГУ. 2010. Серия 10. Вып. 1. С. 95–104.

10. *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.
11. *Тулупьев А.Л.* Байесовские сети доверия и алгебраические байесовские сети: сравнительный анализ выразительной мощности // Информационные технологии и интеллектуальные методы. 1997. Вып. 2. С. 121–147.
12. *Тулупьев А.Л.* Байесовские сети: логико-вероятностный вывод в циклах. СПб.: Изд-во С.-Петербургского ун-та, 2008. 140 с. (Элементы мягких вычислений).
13. *Тулупьев А.Л.* Непротиворечивость оценок вероятностей в алгебраических байесовских сетях. Вестник СПбГУ. Сер. 10. 2009. Вып. 3. С. 144–151.
14. *Тулупьев А.Л.* Непротиворечивость оценок вероятностей в идеалах конъюнктов и дизъюнктов. Вестник СПбГУ. Сер. 10. 2009. Вып. 2. С. 121–131.
15. *Тулупьев А.Л.* Оценка чувствительности результата априорного логико-вероятностного вывода в интеллектуальных информационных системах // Известия высших учебных заведений: Приборостроение. 2009. №9. С. 3–6.
16. *Тулупьев А.Л.* Преобразование ациклических байесовских сетей доверия в алгебраические байесовские сети // Известия высших учебных заведений: Приборостроение. 2009. № 3. С. 21–23.
17. *Тулупьев А.Л.* Согласованность данных и оценка вероятности альтернатив в цикле стохастических предпочтений // Известия высших учебных заведений: Приборостроение. 2009. № 7. С. 3–8.
18. *Тулупьев А.Л., Николенко С.И., Сироткин А.В.* Байесовские сети: логико-вероятностный подход. СПб.: Наука, 2006. 607 с.
19. *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.
20. *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.
21. *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.
22. *Фильченков А.А.* Алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик // Труды СПИИРАН. 2010. Вып. 1 (12). С. 119–133.
23. *Фильченков А.А.* Алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик-собственников // Труды СПИИРАН. 2010. Вып. 3 (14) С. 150–169.
24. *Фильченков А.А.* Алгоритм построения множества минимальных графов смежности при помощи клик владений // Труды СПИИРАН. 2010. Вып. 1 (13). С. 119–133.
25. *Фильченков А.А.* Алгоритм построения множества минимальных графов смежности при помощи клик-собственников владений // Труды СПИИРАН. 2010. Вып. 4 (15). С. 193–212.
26. *Фильченков А.А., Тулупьев А.Л.* Анализ циклов в минимальных графах смежности алгебраических байесовских сетей // Труды СПИИРАН. 2011. Вып. 17 [в печати].
27. *Фильченков А.А., Тулупьев А.Л.* Понятие торакса в применении к исследованию графов смежности алгебраических байесовских сетей // Труды СПИИРАН. 2011. Вып. 16. С. 186–205.

28. *Фильченков А.А., Тулупьев А.Л.* Структурный анализ систем минимальных графов смежности Труды СПИИРАН. 2009. Вып. 11. С. 104–127.
29. *Фильченков А.А., Тулупьев А.Л., Сироткин А.В.* Компаративный анализ клик минимальных графов смежности алгебраических байесовских сетей // Труды СПИИРАН. 2010. Вып. 2 (13). С. 87–105.
30. *Фильченков А.А., Тулупьев А.Л., Сироткин А.В.* Мощность множества минимальных графов смежности // Труды СПИИРАН. 2010. Вып. 4 (15). С. 136–161.
31. *Фильченков А.А., Тулупьев А.Л., Сироткин А.В.* Особенности анализа вторичной структуры алгебраической байесовской сети // Труды СПИИРАН. 2010. Вып. 1 (12). С. 97–118.
32. *Фильченков А.А., Тулупьев А.Л., Сироткин А.В.* Ребра графов смежности в контексте компаративного анализа клик минимальных графов смежности алгебраических байесовских сетей // Труды СПИИРАН. 2010. Вып. 3 (14). С. 132–149.
33. *Фильченков А.А., Тулупьев А.Л., Сироткин А.В.* Структурный анализ клик минимальных графов смежности // Вестник Тверского государственного университета. Сер. Прикладная математика. 2011. Вып. 2.
34. *Юсупов Р.М.* Анализ русской разговорной речи // Вестник Российской академии наук. 2008. Т. № 79. Вып. 3. С. 271–272.
35. *Юсупов Р.М., Ронжин А.Л.* От умных приборов к интеллектуальному пространству // Вестник Российской академии наук. 2010. Том № 80. Вып. 1. С. 45–51.
36. *Юсупов Р.М. Тохтабаев А., Скормин В., Долгих А., Тукеев У., Алтайбек А.* Распознавание механизмов распространения сетевых червей с использованием модели цветных сетей Петри // Проблемы информационной безопасности. Компьютерные системы. 2008. Вып. 3. С. 80–99.
37. *Gorodetsky V.I., Drozdgin V.V., Jusupov R.M.* Application of Attributed Grammar and Algorithmic Sensitivity Model for Knowledge Representation and Estimation // Artificial Intelligence and Information, Control Systems of ROBOTSA. Amsterdam: Elsevier Science Publishers B. V., 1984, P. 232–237.

Фильченков Андрей Александрович — аспирант кафедры информатики математико-механического факультета С.-Петербургского государственного университета (СПбГУ), младший научный сотрудник лаборатории теоретических и междисциплинарных проблем информатики СПИИРАН. Область научных интересов: автоматическое обучение вероятностных графических моделей. Число научных публикаций — 23. aaafil@mail.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. Научный руководитель — А.Л. Тулупьев.

Filchenkov Andrey Alexandrovich — PhD student of Computer Science Department, SPbSU, junior researcher, Theoretical and Interdisciplinary Computer Science Laboratory, SPIIRAS Research area: machine learning of probabilistic graphical models. The number of publications — 23. aaafil@mail.ru, SPIIRAS, 14-th line V.O., 39, St. Petersburg, 199178, Russia; office phone +7(812)328-3337, fax +7(812)328-4450. Scientific advisor — A.L. Tulupyev.

Рекомендовано ТиМПИ СПИИРАН, зав. лаб. А.Л. Тулупьев, д.ф.-м.н., доцент.

Работа поступила в редакцию 08.07.2011.

Поддержка исследования. Работа выполнена при финансовой поддержке РФФИ, проект № **09-01-00861**-а «Методология построения интеллектуальных систем поддержки

принятия решений на основе баз фрагментов знаний с вероятностной неопределенностью», а также гранта Правительства Санкт-Петербурга для победителей конкурса грантов Санкт-Петербурга для студентов, аспирантов, молодых ученых, молодых кандидатов наук 2010 г., диплом ПСП№10697.

РЕФЕРАТ

Фильченков А.А. **Алгоритмы построения третичной структуры алгебраической байесовской сети.**

Алгебраические байесовские сети (АБС) представляют собой вероятностную графическую модель баз фрагментов знаний с неопределенностью. Первичной структурой АБС называется набор фрагментов знаний, а вторичной — граф, построенный над первичной структурой. Третичная структура АБС, представляемая направленным графом полных подграфов, который также называется графом клик, требуется для построения вторичной структуры, а также для анализа первичной структуры.

Доказано, что для данной первичной структуры АБС существует и при том единственная третичная структура АБС, что характеризует граф клик как инвариант с точки зрения различных вторичных структур АБС, возможных к построению над данной первичной структурой.

Алгоритм построения графа клик при помощи потомков формирует граф клик следующим образом: сначала строятся сами клики (строится подмножество вершин, входящих в клики), затем для каждой клики строятся все ее потомки, то есть те клики, в которые можно попасть из этой данной клики. Далее из этого множества отсекаются те клики, в которые можно попасть из потомков данной клики, таким образом, во множестве остаются только те клики, которые соединены ребром с данной (ее сыновья). Во время отбора сыновей, вершины потомков добавляются к множеству вершин данной клики. После работы алгоритма для каждой клики будет сформировано множество ее вершин и множество ее сыновей.

Алгоритм построения графа клик снизу—вверх так же, как и предыдущий, сначала строит сами клики, затем, перебирая клики по убыванию индекса их веса (это гарантирует, что сын будет выбран до родителя) для каждой клики строит множество ее сыновей путем перебора клики с бо̀льшим индексом веса и отсеиванием тех клик, которые являются потомками уже найденных сыновей. К множеству вершин клики добавляются множества вершин ее сыновей. После работы алгоритма для каждой клики будет сформировано множество ее вершин и множество ее сыновей.

Время работы обоих алгоритмов оценено через ряд параметров: числа потомков, предков, сыновей и родителей каждой клики, числа принадлежащих ей вершин и числа ребер максимального графа смежности, вес которых в точности равен весу клики. Для обобщения оценки требуется провести анализ графа клик на предмет установления зависимости этих параметров друг от друга.

Существуют ряд первичных структур, на которых первый алгоритм работает быстрее второго, однако для остальных структур второй алгоритм будет справляться лучше. Так, первый алгоритм эффективен только для построения тех графов клик, у которых небольшая высота и большое число ребер, тогда как в остальных случаях второй алгоритм предпочтительнее.

ABSTRACT

Filchenkov A.A. **Algebraic Bayesian Network Tertiary Structure Synthesis Algorithms.**

Algebraic Bayesian networks (ABN) is a probabilistic graphical model of bases of knowledge patterns with uncertainty. ABN primary structure is a set of the knowledge pattern, and ABN secondary structure is a graph on the primary structure. The ABN tertiary structure represented by directed graph of complete subgraphs, also called a graph cliques is required to synthesize the secondary structure as well as to analyze the primary structure.

It is proven that there are the only and unique ABN tertiary structure on given primary structure, that characterizes the clique graph as the invariant in terms of various ABN secondary structures that are possible on the given primary structure.

The clique graph synthesis descendants algorithm forms the graph cliques as follows: first it constructs cliques themselves (subset of vertices belonging to the clique), then it constructs for each the clique being built all its descendants, that are, those cliques, which can be reached from that clique. Next. cliques of this set are discarded such that can be reached from the descendants of the clique, thus, the set will contain only cliques that are connected by an edge with this (its sons). During the selection of the sons, the vertices belonging to the descendants are added to this clique vertex set. Each clique vertices set and son set will have been constructed when algorithm work is done.

The clique graph synthesis bottom-up algorithm, just like the previous one, first constructs cliques themselves, then it constructs each clique sons set by iterating through cliques with bigger weight index and screening out cliques that are descendants of the clique that have already been identified to be the clique sons. The algorithm do it for each clique by iterating through the clique set from the biggest to the smallest weight index (this ensures that the son will be processed before its parent). Vertices sets of the clique descendants are added to its vertices set. Each clique vertices set and son set will have been constructed when algorithm work is done.

The both algorithms complicities are estimated by the number of parameters: the numbers of each clique descendants, of its ancestors, of its sons and parents, the number of vertices belonging to the clique and the maximum number of edges of the join graph, which weigh in of accuracy equal to the weight clique. To generalize the estimation clique graph analysis is required to find dependences of these parameters on each other.

There are a number of primary structures on which the first algorithm works faster than the second one, but for the other structures the second algorithm will work faster. Thus, the first algorithm is only effective for the synthesis of the such clique graphs, in which has a small height and a large number of edges, while in other cases the second algorithm is preferred.