УДК 004.8

А.А. Фильченков

АЛГОРИТМЫ ПОСТРОЕНИЯ ЭЛЕМЕНТОВ ТРЕТИЧНОЙ ПОЛИСТРУКТУРЫ АЛГЕБРАИЧЕСКОЙ БАЙЕСОВСКОЙ СЕТИ

Фильченков А.А. **Алгоритмы построения элементов третичной полиструктуры алгебраической байесовской сети.**

Аннотация. Роль третичной полиструктуры алгебраической байесовской сети (АБС) заметно возросла. Вводимая изначально в качестве вспомогательного объекта для построения вторичной структуры, третичная полиструктура нашла свое применение в анализе цикличности вторичной структуры без ее непосредственного построения и предполагается к использованию для глобального вывода в АБС. Цель работы — выделение (с последующей систематизацией и оценкой сложности) существующих алгоритмов построения элементов третичной полиструктуры из алгоритмов построения вторичной структуры. В работе рассмотрены существующие алгоритмы построения элементов третичной полиструктуры и оценено время их работы. Приводятся четыре алгоритма построения пустого графа над подмножествами значимых клик и два алгоритма построения родительского графа над множеством стереоклик.

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

Filchenkov A.A. **Algebraic Bayesian Network Tertiary Polystructure Elements Synthesis Algorithms.**

Abstract. The role of algebraic Bayesian network (ABN) polystructure has increased significantly. It had been originally introduced as an auxiliary object for the secondary structure synthesis, but then the tertiary polystructure has found its use in the analysis of secondary structure cyclicity without its direct synthesis. Now the tertiary polystructure is expected to be used for global inference in ABN. The goal of the work is the selection (and subsequent systematization and complicity estimation) the existing algorithms for tertiary polystructure elements synthesis. The existing algorithms for tertiary polystructure elements synthesis are overviewed and the algorithm complicity is estimated in the paper. Four algorithms for synthesizing the empty graph over useful cliques subsets and two algorithms for synthesizing the parent graph over a stereoclique set are presented.

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

1. Введение. Алгебраические байесовские сети (АБС) были предложены в 1993 году [1] как новая парадигма экспертных систем. АБС [7, 8, 9, 13, 21, 24, 26] направлена на решение двух основных проблем в области экспертных систем: дефицита знаний и дефицита математических моделей для представления знаний с неопределенностью [24, 26, 43]. Заметный вклад в преодолении двух указанных видов дефицита внес класс логико-вероятностных графических моделей систем знаний с неопределенностью, которыми и являются АБС [18, 24].

АБС позволяет работать как со скалярными оценками вероятности, так и с интервальными оценками (что является отличительной особенностью АБС) [8, 13, 24]. В первом случае АБС представляет вероятностное распределение над набором атомарных пропозициональных формул (атомов), над которым строится АБС, а во втором — семейство вероятностных распределений.

АБС позволяет поддерживать непротиворечивость сети [19, 20], а также осуществлять априорный и апостериорный логико-вероятностные выводы [5, 6, 10, 14, 21, 22, 24]. Задача априорного вывода состоит в оценке вероятности формулы, построенной над подмножеством алфавита, над которым построена сеть. Первая задача апостериорного вывода состоит в оценке вероятности поступившего свидетельства (которое может быть детерминированным, стохастическим и неточным) для заданной сети. Вторая задача апостериорного вывода состоит в оценке вероятностного распределения, задаваемого сетью, при условии поступившего свидетельства [11, 26].

Одним из основных принципов вероятностных графических моделей является декомпозируемость. Рассуждая об АБС как о представлении вероятностного распределения (или семейства таких распределений), саму АБС можно представить в виде всеобъемлющего фрагмента знаний (ФЗ), построенного над множеством атомов, где под фрагментом знаний понимается идеал положительно означенных конъюнкций за исключением пустого конъюнкта, называемый идеалом конъюнктов с заданным вероятностным распределением на его элементах. Однако хранение и обработка такого объема данных нецелесообразна, поэтому всеобъемлющий ФЗ допускает декомпозицию на более маленькие фрагменты знаний [10, 24, 25]. При этом, чтобы представить имеющиеся между фрагментами знаний, придется использовать графы смежности [26]. В этом случае речь уже будет идти о поддержке непротиворечивости и проведении логико-вероятностного вывода над этим графом (эти операции называются глобальными в противоположность операциям, проводимым в одном фрагменте знаний, которые в свою очередь называются локальными). Таким образом, алгоритмизация всех глобальных операций АБС опирается на граф смежности [7], что делает его чрезвычайно важным для теории АБС [16]. Набор фрагментов знаний называют первичной структурой АБС, а граф, построенный над ним, — вторичной структурой АБС. Обычно в этом контексте говорят о максимальных фрагментах знаний (МФЗ), указывая на то, что ни один МФЗ не содержится полностью ни в каком другом МФЗ.

Среди прочих выделяют минимальные по числу ребер графы смежности [4, 24, 27, 34, 35, 38], которые обладают рядом особенностей, в том числе и следующей: если над данной первичной структурой возможно построение ациклических графов смежности, то множество ациклических графов смежности будет совпадать с множеством минимальных графов смежности [33]. Необходимо отметить, что ацикличность графа является одним из условий для применения известной на данный момент алгоритмизации поддержания глобальной непротиворечивости и осуществления глобального априорного вывода [12, 15, 16, 23]. Это обусловило интенсивные исследования в отношении минимальных графов смежности и множества минимальных графов смежности. Так, известны алгоритмы построения множества минимальных графов смежности [27–31, 35] и произвольного элемента этого множества [3].

Все эти алгоритмы требуют построения элементов третичной полиструктуры АБС [36] — направленных графов, которые строятся над множеством подграфов максимального графа смежности. Изначально элементы третичной полиструктуры рассматривались лишь как вспомогательные построения для исследования минимальных графов смежности (и графов смежности в целом) [27, 35]. Однако, как показали дальнейшие исследования и попытки алгоритмизации ряда процессов, роль третичной полиструктуры оказалась заметно более серьёзной. Сначала третичная полиструктура стала применяться для выявления ацикличности вторичной структуры без ее непосредственного построения [33]. Сейчас является актуальным вопрос о возможности осуществления глобального логико-вероятностного вывода на третичной полиструктуре без использования вторичной структуры.

Кроме того, хотя известны алгоритмы построения элементов третичной полиструктуры [32], существует множество алгоритмов построения вторичной структуры, которые требуют построения вспомогательных элементов третичной полиструктуры [3, 27–31, 35].

Цель данной работы — выделение (с последующей систематизацией и оценкой сложности) существующих алгоритмов построения элементов третичной полиструктуры из алгоритмов построения вторичной структуры.

2. Основные определения и обозначения. Перед тем, как рассмотреть алгоритмы построения элементов третичной полиструктуры АБС, определим те понятия, которыми будем пользоваться.

2.1. Базовые определения. Будем следовать обозначениям, введенным в работах [35, 39].

Как и сама АБС, третичная полуструктура 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. Понятие «полиструктура» используется для того, чтобы подчеркнуть, что у АБС есть семейство графов, которые относятся к третичной полиструктуре. Хотя над первичной структурой можно построить множество графов, конкретная сеть представляется только одним графом смежности, тогда как элементы третичной полиструктуры не являются взаимоисключающими и могут одновременно использоваться для различных целей в отношении одной и той же АБС.

Вес $W(V)$ вершины V — множество атомов алфавита, вошедших в соответствующий МФЗ. 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, представляемое в двоичной записи соответствующим весом [27, 35]. Так, например, набор МФЗ $\{ax, ay, xz\}$, заданный, соответственно, над алфавитом $\{a, x, y, z\}$, индексируется следующим образом:

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

Эта индексация естественным образом продолжается до множества всех возможных весов, построенных над соответствующим алфавитом [6, 7, 9, 17, 21, 24, 26].

Вес, равный пересечению весов двух вершин, будем называть *значимым весом* [35].

2.2. Значимые клики и третичная полиструктура. Будем следовать обозначениям, введенным в работе [36].

Значимая клика веса U — полный граф, построенный на максимальном по включению множестве вершин, веса которых содержат значимый вес U. Множество всех значимых клик будем обозначать как Clique.

Вес значимой клики P будем обозначать как $W(P)$. Будем говорить, что значимая клика P *содержит* значимую клику Q, если $W(Q)$ содержит $W(P)$ или, что то же самое, множество вершин значимой клики P содержит множество вершин значимой клики Q. Следует отметить, что любой значимый вес будет являться весом какой-либо значимой клики.

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

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

Родительский граф над подмножеством значимых клик A — направленный граф, вершинами которого являются значимые клики из множества A. Ребро из вершины P в вершину Q проведено, если значимая клика P является родителем значимой клики Q. Будем считать, что сыновья значимой клики упорядочены по индексу их веса.

Пустой граф над подмножеством значимых клик A — граф, построенный над множеством A, не содержащий ребер.

Здесь важно отметить, что пустой граф — это граф, не содержащий ребер [42], т. е. граф, соответствующий набору вершин, над которым он строится.

2.3. Классификация значимых клик и их элементов. Будем следовать обозначениям, введенным в работах [32, 35–37, 39–41].

Собственное ребро значимой клики — ребро, вес которого совпадает с весом этой значимой клики. Значимая клика содержит все собственные ребра [36].

Основное множество вершин значимой клики — множество вершин, являющихся концами собственных ребер значимой клики. Каждая вершина, входящая в значимую клику, входит в множество основных вершин либо самой значимой клики, либо в множество основных вершин какого-либо ее потомка [32].

Строгое сужение на вес u — граф, получающийся удалением всех ребер веса u из значимой клики веса u [35].

Любая значимая клика является полным графом, тогда как строгое сужение может распадаться на компоненты связности [35]. Такие компоненты называются *владениями*. Известна теорема о классификации

владений [41], которая утверждает, что любое владение является либо доменной вершиной, либо вассалом, либо братством, где

- *доменная вершина* — подграф, состоящий ровно из одной вершины, которая принадлежит значимой клике, но не принадлежит никакому ее сыну;
- *вассал* — подграф, вершины которого совпадают с вершинами, которые содержит какой-либо из сыновей значимой клики.
- *братство* — это максимальный по включению набор сыновей значимой клики, такой, что любые два сына из этого набора связны в смысле пересечения сыновей по вершинам.

Клики делятся, в свою очередь, на несколько классов [37, 40].

Биклика — значимая клика, состоящая ровно из двух вершин. Множество всех биклик будем обозначать как Biclique.

Значимая моноклика — значимая клика веса U, такая, что строгое сужение на вес U (см. 2.3) связно. Множество значимых моноклик будем обозначать как Monoclique. Значимые моноклики будем также разделять на *моноклики-*1 и *моноклики-n*. Первая отличается тем, что у нее ровно одно собственное ребро, тогда как у второй — более одного.

Стереоклика — значимая клика, не являющаяся бикликой или значимой монокликой. Множество всех стереоклик будем обозначать как Stereoclique.

Следует отметить, что

$$\text{Clique} = \text{Stereoclique} \cup \text{Biclique} \cup \text{Monoclique}.$$

Дополнительно введем множество

$$\text{NotBiclique} = \text{Stereoclique} \cup \text{Monoclique} = \text{Clique} \backslash \text{Biclique}.$$

2.4. Соглашения об оформлении алгоритмов. 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 $O(1)$.

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, Possessions, Domains могут быть организованы, например, хэш-таблицей [2], предоставляя доступ к своим элементам по ключам, соответствующим индексам веса этих элементов (вместо $\text{Vertecies}[I(u)]$ мы будем писать просто $\text{Vertecies}[u]$). 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.

Рассматриваемые в работе алгоритмы возвращают либо пустой, либо родительский граф над каким-либо подмножеством множества значимых клик.

Собственное подмножество значимых клик $C \subset \mathrm{Clique}$ будем представлять парой $\langle \mathrm{Vertecies}, \mathrm{CWeights} \rangle$, где Vertecies — упорядоченное множество, такое, что для любого значимого веса u $\mathrm{Vertecies}[u]$ — множеств вершин, входящих в значимую клику веса u, а AWeights — индексирующее множество значимых весов:

$$C = \{\mathrm{Vertecies}[u] | u \in \mathrm{CWeights}\}.$$

Фактически, мы будем представлять подмножество как базовое множество и множество индексов элементов этого множества, которые входят в подмножество.

Родословный граф над подмножеством C будем представлять не традиционной парой $\langle C, E \rangle$, а тройкой $\langle \mathrm{Vertecies}, \mathrm{CWeights}, \mathrm{Sons} \rangle$, где Sons — упорядоченное множество, такое, что для любого значимого веса u $\mathrm{Sons}[u]$ — множество сыновей значимой клики веса u:

$$\langle C, E \rangle = \langle C, \{(C[u], C[w]) | w \in \mathrm{Sons}[u]\} \rangle =$$
$$= \langle \{\mathrm{Vertecies}[u] | u \in \mathrm{CWeights}\},$$
$$\{(\mathrm{Vertecies}[u], \mathrm{Vertecies}[w]) | w \in \mathrm{Sons}[u], u \in \mathrm{CWeights}\} \rangle.$$

В записи алгоритмов

- запись «$a \leftarrow b$» обозначает присвоение значения элемента b элементу a;
- запись «$S \rightarrow e\colon \mathrm{Cond}(e)$» обозначает извлечение произвольного элемента, удовлетворяющего условиям Cond, из множества S в переменную e.

Корректность работы всех приводимых алгоритмов следует из корректности работы тех алгоритмов построения множества минимальных графов смежности или произвольного минимального графа смежности, частью которых алгоритмы построения элементов третичной полиструктуры и являлись. Соответствующие алгоритмы и доказательства приводятся в [3, 27–31, 35].

Через $\textsc{Is\_Connected}(V, E)$ будем обозначать булевозначную функцию, которая возвращает TRUE, если граф $\langle \mathrm{V}, \mathrm{E} \rangle$ связен и FALSE в обратном случае, а через $\textsc{Components}(V, E)$ обозначать функцию, которая возвращает множество компонент связности графа $\langle V, E \rangle$. Обе функции имеют одинаковую сложность и могут быть реализованы различными способами. Мы будем исходить из алгоритма, использующего

систему непересекающихся множеств [2], приняв его оценку времени работы за $O(|V|\log|V|)$.

В работе мы будем, насколько это возможно, приводить точную, а не асимптотическую оценку сложности для того, чтобы иметь возможность в дальнейшем за счет выявления соотношений между исходными переменными уточнить такие оценки.

3. Алгоритм построения пустого графа над множеством Clique. В ряде ранних работ по исследованию минимальных графов смежности [3, 27, 35] для построения множества минимальных графов смежности или произвольного минимального графа использовалось построение множества значимых клик (которое соответствует пустому графу над данным множеством как объекту третичной полиструктуры). Алгоритм построения этого графа приведен на листинге 1.

```
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:       UsefulWeighs ← UsefulWeighs ∪ {w}
 6:       Vertecies[w] ← ∅
 7:    end if
 8: end for
 9: for all w ∈ Weights do
10:    for all u ∈ UsefulWeights
11:       if u ⊂ w do
12:          Vertecies[u] ← Vertecies[u] ∪ {w}
13:       end if
14:    end for
15: end for
16: return Vertecies
```

Листинг 1. Алгоритм построения пустого графа над множеством значимых клик.

В цикле (1–7) строится множество значимых весов UsefulWeights путем перебора всех пар вершин.

В цикле (9–15) для каждого значимого веса строится множество вершин, попавших в соответствующую клику путем полного перебора всех вершин.

Утверждение 1. Алгоритм построения пустого графа над множеством значимых клик работает за не более чем

$$2|W|^2 + \sum_{u \in U} (2\log_2|U| \cdot e_u + |W| + 2|V_u| \cdot \log_2|V_u|),$$

где W — набор МФЗ, U — множество значимых весов, e_u — число пар вершин, пересечение весов которых в точности равно u; V_u — множество вершин, входящих в значимую клику веса u.

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

$$2|W|^2 \quad (1)$$

поскольку две операции будут осуществляться для каждой пары вершин.

На добавление значимого веса на шаге (5) всего потребуется не более

$$2\log_2|U| \cdot \sum_{u \in U} e_u, \quad (2)$$

т. к. оно 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 добавление элемента в упорядоченное множество U.

На выполнение проверки на шаге (11) в сумме потребуется

$$|U| \cdot |W|. \quad (3)$$

потому что она осуществляется для всех пар значимых весов и вершин.

На выполнение шага (12) в сумме потребуется не более

$$\sum_{u \in U} (2|V_u| \cdot \log_2|V_u|), \quad (4)$$

потому что для каждого веса u строится упорядоченное множество вершин, вошедших в значимую клику соответствующего веса, за не более чем

$$2|V_u| \cdot \log_2|V_u|.$$

Сложив выражения (1)–(4) и сгруппировав сумму относительно u, получим искомое выражение.

4. Алгоритм построения пустого графа над множеством NotBiclique. В работе [35] описывается также улучшенный алгоритм

построения множества минимальных графов смежности, который включает в себя построение пустого графа над множеством значимых клик, из которого исключается множество биклик Biclique. Алгоритм построения этих графа и множества приведен на листинге 2.

```
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:       UsefulWeighs ← UsefulWeighs ∪ {w}
 6:       Vertecies[w] ← ∅
 7:    end if
 8: end for
 9: for all w ∈ Weights do
10:    for all u ∈ UsefulWeights
11:       if u ⊂ w do
12:          Vertecies[u] ← Vertecies[u] ∪ {w}
13:       end if
14:    end for
15: end for
16: NotBiWeights ← ∅
17: for i ← 1 to |UsefulWeights| do
18:    u ← UsefulWeights[i]
19:    if |Vertecies[u]| > 2 then
20:       NotBiWeights ← NotBiWeights ∪ {u}
21:    end if
22: end for
23: return NotBiWeights, Vertecies
```

Листинг 2. Алгоритм построения пустого графа над множеством NotBiclique.

Шаги (1–15) полностью повторяют алгоритм построения пустого графа над множеством значимых сужений (листинг 1).

В цикле (17–22) перебирается все множество значимых весов, и если значимая клика содержит больше двух вершин (условие на шаге (18)), то значимая клика добавляется к множеству NotBiclique.

Утверждение 2. Алгоритм построения пустого графа над множеством NotBiclique работает за не более чем

$$2|W|^2 + 2|U| + |N| +$$
$$+\sum_{u\in U}(2\log_2|U|\cdot e_u + |W| + 2|V_u|\cdot\log_2|V_u|),$$

где W — набор МФЗ, U — множество значимых весов, e_u — число пар вершин, пересечение весов которых в точности равно u; V_u — множество вершин, входящих в значимую клику веса u, N — множество NotBiclique.

Доказательство. Из утверждения 1 известно, что на выполнение шагов (1–15) требуется не более

$$2|W|^2 + \sum_{u\in U}(2\log_2|U|\cdot e_u + |W| + 2|V_u|\cdot\log_2|V_u|). \quad (5)$$

На выполнение шагов (18) и (19) всего потребуется

$$2|U|, \quad (6)$$

поскольку они будет осуществляться для всех значимых весов.

На создание упорядоченного множества NotBiclique на шаге (19) потребуется не более чем

$$|S| + |M|, \quad (7)$$

потому что при создании этого множества веса перебираются по порядку, поэтому добавление одного элемента осуществляется за константу.

Сложив формулы (5)–(7), мы получим искомое выражение.

Замечание 1. Алгоритм построения пустого графа над множеством NotBiclique (листинг 2) работает заведомо не быстрее алгоритма построения пустого графа над множеством Clique (листинг 1).

5. Алгоритм построения пустого графа над множеством стереоклик полным перебором. В работе [28] предложены соображения, которые можно объединить в один принцип:

Принцип 1. Для построения множества минимальных графов смежности достаточно построить множество Stereoclique.

На этом принципе основывается алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик [28], из которого можно выделить алгоритм построения пустого графа над множеством стереоклик полным перебором (листинг 3).

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 ← w_1 ∩ w_2
 4:    if w ≠ ∅ do
 5:       UsefulWeighs ← UsefulWeighs ∪ {w}
 6:       Vertecies[w] ← ∅
 7:    end if
 8: end for
 9: for all w ∈ Weights do
10:    for all u ∈ UsefulWeights
11:       if u ⊂ w do
12:          Vertecies[u] ← Vertecies[u] ∪ {w}
13:       end if
14:    end for
15: end for
16: StereoWeights ← ∅
17: for i ← 1 to |UsefulWeights| do
18:    u ← UsefulWeights[i]
19:    if |Vertecies[u]| > 2 then
20:       iscon ← IsConnected(Vertecies[u],
             {(v_1, v_2)|v_1, v_2 ∈ Vertecies[u], u ⊂ v_1 ∩ v_2})
21:       if iscon then
22:          StereoWeights ← StereoWeights ∪ {u}
23:       end if
24:    end if
25: end for
26: return StereoWeights, Vertecies
```

Листинг 3. Алгоритм построения пустого графа над множеством стереоклик полным перебором.

Алгоритм на шагах (1–15) полностью повторяет алгоритм построения пустого графа над множеством значимых клик. В цикле (2–8) строится множество значимых весов путем попарного перебора всех пар вершин.

В цикле (9–15) для каждого значимого веса строится множество всех вершин, попадающих в соответствующее сужение (содержащих этот вес) путем попарного перебора всех вершин и всех значимых весов.

В цикле (17–25) строится множество весов стереоклик (StereoWeights) за счет исключения всех биклик в условном операторе (19–24) и исключения всех значимых моноклик в условном опера-

торе (21–23). Исключение значимых моноклик осуществляется за счет проверки связности значимых сужений соответствующего веса.

Утверждение 3. Алгоритм построения пустого графа над множеством стереоклик полным перебором работает за не более чем

$$2|W|^2 + 2|U| + |N| + |S| + \sum_{u \in N} O(V_u \cdot \log V_u) + \\ + \sum_{u \in U} (2\log_2|U| \cdot e_u + |W| + 2|V_u| \cdot \log_2|V_u|) \quad (8)$$

где W — набор МФЗ, U — множество значимых весов, e_u — число пар вершин, пересечение весов которых в точности равно u; V_u — множество вершин, входящих в значимую клику веса u, N — множество NotBiclique, S — множество Stereoclique.

Доказательство. Из утверждения 2 известно, что на шаги (1–19) потребуется не более

$$2|W|^2 + 2|U| + \\ + \sum_{u \in U} (2\log_2|U| \cdot e_u + |W| + 2|V_u| \cdot \log_2|V_u|). \quad (9)$$

На выполнение шага (20) потребуется не более

$$\sum_{u \in N} O(V_u \cdot \log V_u), \quad (10)$$

так как будет вызвана для каждой значимой клики из NotBiclique, и будет выполняться для графа над V_u вершинами.

На выполнение проверки на шаге (21) потребуется всего

$$|N|, \quad (11)$$

поскольку она будет выполняться для каждой значимой клики из NotBiclique.

На формирование упорядоченного множества Clique на шаге (22) всего потребуется

$$|S|, \quad (12)$$

потому что веса перебираются упорядоченно, и вставка будет осуществляться за константу.

Сложив формулы (9)–(12), получим искомое выражение.

Замечание 2. Алгоритм построения пустого графа над множеством стереоклик полным перебором (листинг 3) работает заведомо хуже алгоритма построения пустого графа над NotBiclique (листинг 2).

6. Алгоритм построения пустого графа над множеством стереоклик перебором собственных ребер. В работе [29] предложены

соображения, которые можно переформулировать следующим образом:

Принцип 2. Для построения множества вершин, входящих в стереоклику, достаточно построить ее основное множество вершин.

С алгоритмической точки зрения, принцип 2 позволяет строить основное множество вершин сразу при переборе всех ребер максимального графа смежности.

```
Require: Weights
Ensure: ∄w_1, w_2 ∈ Weights: w_1 ⊆ w_2
 1: for all w_1, w_2 ∈ Weights, w_1 ≠ w_2 do
 2:   w ← w_1 ∩ w_2
 3:   if w ≠ ∅ do
 4:     if w ∉ UsefulWeighs do
 5:       UsefulWeighs ← UsefulWeighs ∪ {w}
 6:       Vertecies[w] ← ∅
 7:     end if
 8:     Vertecies[w] ← Vertecies[w] ∪ {w_1}
 9:     Vertecies[w] ← Vertecies[w] ∪ {w_2}
10:   end if
11: end for
12: StereoWeights ← ∅
13: for i ← |UsefulWeights| downto 1 do
14:   u ← UsefulWeigths[i]
15:   if |Vertecies[u]| > 2 do
16:     iscon ← IsConnected(Vertecies[u],
                  {(v_1, v_2)|v_1, v_2 ∈ Vertecies[u], u ⊂ v_1 ∩ v_2})
17:     if is_con then
18:       StereoWeights ← StereoWeights ∪ {u}
19:     end if
20:   end if
21: end for
22: return StereoWeights, Vertecies
```

Листинг 4. Алгоритм построения пустого графа над множеством стереоклик перебором собственных ребер.

На принципах 1 и 2 основывается алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик–собственников [29], из которого можно выделить алгоритм построения

пустого графа над множеством стереоклик перебором собственных ребер (листинг 4).

Шаги алгоритма (13–21) повторяют шаги (17–25) алгоритма построения пустого графа над множеством стереоклик полным перебором.

В цикле (1–11) на основе попарного перебора всех вершин строится множество значимых весов UsefulWeigths и для каждого значимой клики строится на шагах (8–9) ее основное множество вершин путем добавления к нему концов всех ее собственных ребер.

В цикле (13–21) происходит отбор стереоклик за счет исключения биклик **(**на проверке (15)**)** и исключения моноклик **(**на проверке (17)**).**

Утверждение 4. Алгоритм построения пустого графа над множеством стереоклик полным перебором работает за не более чем

$$2|W|^2 + 2|U| \cdot \log_2|U| + 2|U| + |N| + |S| + \sum_{u \in N} O(V_u \cdot \log V_u) + \\ + \sum_{u \in U} (2\log_2|V_u| \cdot e_u + 2\log_2|U| \cdot e_U) \qquad (13)$$

где W — набор МФЗ, U — множество значимых весов, e_u — число пар вершин, пересечение весов которых в точности равно u; V_u — множество вершин, входящих в значимую клику веса u, N — множество NotBiclique, S — множество Stereoclique.

Доказательство. Из [32] известно, что на шаги (1–11) потребуется не более

$$2|W|^2 + 2|U| \cdot \log_2|U| + \\ + \sum_{u \in U} (2\log_2|V_u| \cdot e_u + 2\log_2|U| \cdot e_U) \qquad (14)$$

Из утверждения 2 известно, что на шаги (13–21) потребуется не более

$$2|U| + |N| + |S| + \sum_{u \in N} O(V_u \cdot \log V_u). \qquad (15)$$

Сложив формулы (14)–(15), получим искомое выражение.

7. Сравнение алгоритмов построения пустого графа над множеством стереоклик. Благодаря тому, что в разделах 6 и 7 были сформулированы алгоритмы для построения одного и того же элемента третичной полиструктуры, мы можем их сравнить.

Рассмотрим формулы (8) и (13) (из утверждений 3 и 4 соответственно), которые выражают сложность двух алгоритмов. Вычтем из первой вторую:

$$\left(2|W|^2 + 2|U| + |N| + |S| + \sum_{u \in N} O(V_u \cdot \log V_u)\right) +$$
$$+ \left(\sum_{u \in U} (2\log_2|U| \cdot e_u + |W| + 2|V_u| \cdot \log_2|V_u|)\right) -$$

$$-(2|W|^2 + 2|U| \cdot \log_2|U| + 2|U| + |N| + |S|) -$$
$$-\left(\sum_{u \in N} O(V_u \cdot \log V_u) + \sum_{u \in U} (2\log_2|V_u| \cdot e_u + 2\log_2|U| \cdot e_U)\right).$$

Сократив, получим

$$\left(\sum_{u \in U} (|W| + 2|V_u| \cdot \log_2|V_u| - 2\log_2|V_u| \cdot e_u) - 2|U| \cdot \log_2|U|\right) =$$
$$= \sum_{u \in U} (|W| - 2\log_2|U| + 2|V_u|(\log_2|V_u| - e_u)).$$

Более подробно исследовать эту формулу мешает высокая сложность объектов, которые она характеризует. Рассматривая же покомпонентно, можно обратить внимание на то, что $|W|$ чаще всего больше, чем $2\log_2|U|$, потому что U — множество весов значимых ребер, очевидно, не больше, чем число значимых ребер, которое не больше, чем $|W|^2$, поэтому $2\log_2|U| \leq 2\log_2|W|^2 = 4\log_2|W|$, что оказывается меньше $|W|$ уже при $|W| > 17$. В то же время второй компонент так же может быть как положительным, так и отрицательным. Положительным он может быть только в том случае, если в значимой клике веса u почти все (а именно более $|V_u|^2 - \log_2|V_u|$) ребра принадлежат ее сыновьям. Это, в свою очередь, зависит от размера и структуры клик. При этом следует отметить, что эта разность домножается на число вершин соответствующей значимой клики, так что итоговое произведение может легко превосходить общее число вершин, над которыми строится максимальный граф смежности.

Однако, в целом, авторы склонны полагать, что в значительной части случаев, особенно для большого числа вершин, алгоритм построения родительского графа над множеством стереоклик полным перебором работает быстрее алгоритма построения множества стереоклик перебором собственных ребер. Открытым остается вопрос о том, каков порядок выигрыша в случае использования первого алгоритма.

В заключение раздела приведем два примера.

Рис. 1. Граф смежности, представляющий цепь из n вершин, состоящих из двух атомов.

На рис. 1 приведен пример графа, для которого родительский граф над множеством стереоклик будет строиться быстрее при помощи алгоритма построения перебором значимых ребер. Действительно, для этого графа

$$\sum_{u\in U}(|W| - 2\log_2|U| + 2|V_u|(\log_2|V_u| - e_u)) =$$
$$= \sum_{i=1}^{n-1}\left(n - 2\log_2(n-1) + 2\cdot 2\cdot(\log_2 2 - 1)\right) =$$
$$= (n-1)\cdot(n - 2\log_2(n-1)),$$

что положительно при всех натуральных n (больших 1).

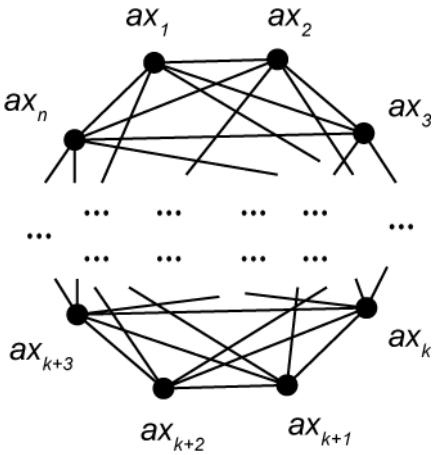

Рис. 2. Граф смежности, представляющий полный граф из n вершин, состоящих из двух атомов.

С другой стороны, на рис. 2 приведен пример графа, для которого родительский граф над множеством стереоклик будет строится быстрее при помощи алгоритма построения полным перебором. Для этого графа

$$\sum_{u\in U}(|W| - 2\log_2|U| + 2|V_u|(\log_2|V_u| - e_u)) =$$
$$= (n - 2\log_2 1 + 2n(\log_2 n - n^2)) =$$
$$= (n + 2n\log_2 n - n^2)),$$

что отрицательно при всех натуральных n (больших 1).

8. Алгоритм построения родительского графа над множеством стереоклик полным перебором. В работе [30] предложен принцип

улучшения алгоритма построения множества минимальных графов смежности при помощи клик–собственников, который можно переформулировать следующим образом:

Принцип 3. Поиск числа владений значимой клики нужно осуществлять через поиск компонент связности на графе, построенном над сыновьями значимой клики, ребра которого соединяют пересекающихся сыновей.

На принципах 1 и 3 основывается алгоритм построения множества минимальных графов смежности при помощи клик владений [30], из которого можно выделить алгоритм построения пустого графа над множеством стереоклик (листинг 4).

```
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:       UsefulWeighs ← UsefulWeighs ∪ {w}
 6:       Vertecies[w] ← ∅
 7:    end if
 8: end for
 9: for all w ∈ Weights do
10:    for all u ∈ UsefulWeights
11:       if u ⊂ w do
12:          Vertecies[u] ← Vertecies[u] ∪ {w}
13:       end if
14:    end for
15: end for
16: StereoWeights ← ∅
17: for i ← |UsefulWeights| downto 1 do
18:    u ← UsefulWeigths[i]
19:    Domains[u] ← Vertecies[u]
20:    Sons[u] ← ∅
21:    if |Domains[u]| > 2 then
22:       for j ← |UsefulWeights| downto i + 1 do
23:          v ← UsefulWeigths[j]
24:          if u ⊂ v then
25:             Sons[u] ← Sons[u]\Sons[v]
```

```
26:            Sons[u] ← Sons[u] ∪ {v}
27:            Domains[u] ← Domains[u]\Domains[v]
28:          end if
29:        end for
30:        BrotherPairs ← ∅
31:        for all s1, s2 ∈ Sons[u], s1 ≠ s2 do
32:          if Vertecies[s1] ∩ Vertecies[s2] ≠ ∅ do
33:            BrotherPairs ← BrotherPairs ∪ {(s1, s2)}
34:          end if
35:        end for
36:        Possessions[u] ← COMPONENTS(Sons[u], BrotherPairs)
37:        Possessions[u] ← Possessions[u] ∪ Domains[u]
38:        if |Possessions[u]| > 1 then
39:          StereoWeigths ← StereoWeighs ∪ {u}
40:        end if
41:      end if
42:    end for
43:    return StereoWeights, Vertecies, Sons, Possessions
```

Листинг 5. Алгоритм построения родительского графа над множеством стереоклик полным перебором.

Шаги (1–15) алгоритма совпадают с шагами (1–15) алгоритма построения пустого графа над множеством стереоклик полным перебором.

В цикле (2–8) строится множество значимых весов UsefulWeight, а в цикле (9–15) для каждой клики строится множество вершин, которые в нее попали.

В цикле (17–42) последовательным перебором весом (поднимаясь по родительскому графу от сыновей к родителям) отсеиваются биклики и моноклики, для стереоклик и моноклик строятся множества владений. Проверка на шаге (21) отсеивает биклики. Далее в цикле (22–29) для всех весов, индекс которых больше рассматриваемого на текущей итерации цикла, на шаге (24) проверяется, являются ли они потомками значимой клики рассматриваемого веса, и если являются, то добавляются к множеству детей значимой клики, их сыновья удаляются из множества сыновей рассматриваемой клики, а доменные вершины удаляются из множества доменных вершин рассматриваемой значимой клики (которое на шаге (19) было инициализировано вершинами рассматриваемой значимой клики). В цикле (31–35) строится мно-

жество пар пересекающихся сыновей рассматриваемой клики. На шаге (36) вычисляются компоненты связности получившегося графа, которые являются не доменными владениями клики. Доменные вершины изначально не включаются в граф, потому что они заведомо являются компонентами связности. Вместо этого они добавляются ко множеству владений на шаге (37). Проверка на шаге (38) отсеивает моноклики.

Утверждение 5. Алгоритм построения родительского графа над множеством стереоклик полным перебором работает за не более чем

$$2|W|^2 + 3|U| + |N|^2 + 3|N| + |S| + \\ + \sum_{u \in N} |D_u| + 2|V_u| \cdot \log_2 |V_u| + \\ + \sum_{u \in U} (2 \log_2 |U| \cdot e_u + |W| + 2|V_u| \cdot \log_2 |V_u|) + \\ + O\left(\sum_{u \in N} \left(|B_u| + |S_u|^2 + \log |S_u| \cdot \sum_{v \in D_u} |P_v| \right) \right),$$

где W — набор МФЗ, U — множество значимых весов, e_u — число пар вершин, пересечение весов которых в точности равно u; V_u — множество вершин, входящих в значимую клику веса u, N — множество NotBiclique, S — множество Stereoclique, S_u — множество сыновей значимой клики веса u, D_u — множеством потомков значимой клики веса u, P_u — множество родителей значимой клики веса u.

Доказательство. Из утверждения 1 известно, что на шаги (1–15) потребуется не более

$$2|W|^2 + \sum_{u \in U} (2 \log_2 |U| \cdot e_u + |W| + 2|V_u| \cdot \log_2 |V_u|). \quad (16)$$

На выполнение шагов (18, 19, 21) суммарно потребуется

$$3|U|, \quad (17)$$

так как они будут выполняться для всех значимых весов.

На выполнение шагов (23–24) суммарно потребуется

$$|N|^2 + |N|, \quad (18)$$

поскольку они будут выполняться для всех упорядоченных пар $\langle i, j \rangle$, где i, j — индексы значимых весов из NotBiClique.

На выполнение шага (25) потребуется

$$O\left(\sum_{u \in N} \left(\log |S_u| \cdot \sum_{v \in D_u} |P_v| \right) \right), \quad (19)$$

потому что сыновья каждого потомка значимой клики веса u, принадлежащей NotBiClique, будут вычитаться из Sons. Если заметить, что сыновья каждого потомка также являются потомками, то каждый потомок веса v будет вычтен столько раз, сколько у него предков, являющихся потомками значимой клики веса u, то есть не более чем P_v раз.

На выполнение шага (26) потребуется

$$\sum\nolimits_{u \in N} |D_u|, \tag{20}$$

потому все предки значимой клики веса u, принадлежащей NotBiClique, будут последовательно добавлены в это множество.

На выполнение шага (27) потребуется не более

$$\sum_{u \in N} (2|V_u| \cdot \log_2 |V_u|), \tag{21}$$

так как объединение доменных вершин всех потомков является подмножеством доменных вершин значимой клики.

На выполнение проверки на шаге (32) потребуется не более

$$\sum\nolimits_{u \in N} |S_u|^2, \tag{22}$$

так как выполняется для всех пар сыновей для каждой значимой клики из NotBiClique.

На выполнение шага (33) потребуется

$$\sum\nolimits_{u \in N} |B_u|, \tag{23}$$

где B_u — множество пар сыновей значимой клики веса u, имеющих общие вершины, так как он происходит для всех пар пересекающихся сыновей.

На выполнение вычисления на шаге (36) потребуется

$$\sum\nolimits_{u \in N} O(|S_u| \log |S_u|), \tag{24}$$

потому что будет выполняться для всех значимых клик, выделяя компоненты связности у графа, вершинами которого являются сыновья значимой клики.

На выполнение шагов (37) и (38) потребуется

$$2|N|. \tag{25}$$

На выполнение шага (39) потребуется

$$|S|, \tag{26}$$

потому что веса добавляются к упорядоченному множеству последовательно.

Сложив формулы (16)–(26) и заметив, что

$$|S_u| \log |S_u| = o(|S_u|^2),$$
$$|B_u| \leq |S_u|^2,$$

получим искомое выражение

9. Алгоритм построения родительского графа над множеством стереоклик перебором собственных ребер. В работе [31] предложен алгоритм построения множества минимальных графов смежности при помощи клик–собственников владений, который основывается одновременно на принципах 1, 2 и 3. Из него можно выделить алгоритм построения родительского графа над множеством стереоклик перебором собственных ребер (листинг 6).

```
Require: Weights
Ensure: ∄w_1, w_2 ∈ Weights: w_1 ⊆ w_2
 1: for all w_1, w_2 ∈ Weights, w_1 ≠ w_2 do
 2:    w ← w_1 ∩ w_2
 3:    if w ≠ ∅ do
 4:       if ∄Vertecies[w] do
 5:          Vertecies[w] ← ∅
 6:       end if
 7:       Vertecies[w] ← Vertecies[w] ∪ {w_1}
 8:       Vertecies[w] ← Vertecies[w] ∪ {w_2}
 9:    end if
10: end for
11: StereoWeights ← ∅
12: for i ← |UsefulWeights| downto 1 do
13:    u ← UsefulWeigths[i]
14:    Domains[u] ← Vertecies[u]
15:    Sons[u] ← ∅
16:    if |Domains[u]| > 2 then
17:       for j ← |UsefulWeights| downto i + 1 do
18:          v ← UsefulWeigths[j]
19:          if u ⊂ v then
20:             Sons[u] ← Sons[u]\Sons[v]
21:             Sons[u] ← Sons[u] ∪ {v}
22:             Domains[u] ← Domains[u]\Domains[v]
23:          end if
```

```
24:       end for
25:       BrotherPairs ← ∅
26:       for all s_1, s_2 ∈ Sons[u], s_1 ≠ s_2 do
27:         if Vertecies[s_1] ∩ Vertecies[s_2] ≠ ∅ do
28:           BrotherPairs ← BrotherPairs ∪ {(s_1, s_2)}
29:         end if
30:       end for
31:       Possessions[u] ← COMPONENTS(Sons[u], BrotherPairs)
32:       Possessions[u] ← Possessions[u] ∪ Domains[u]
33:       if |Vassals[u]| > 1 then
34:         StereoWeigths ← StereoWeighs ∪ {u}
35:       end if
36:     end if
37: end for
38: return StereoWeights, Vertecies, Sons, Possessions
```

Листинг 6. Алгоритм построения родительского графа над множеством стереоклик перебором собственных ребер.

Строки алгоритма (1–11) совпадают со строками (1–11) алгоритма построения пустого графа над множеством Stereoclique перебором значимых весов (листинг 4), а строки алгоритма (12–38) совпадают со строками (17–43) алгоритма построения родительского графа над множеством Stereoclique полным перебором (листинг 5).

Утверждение 6. Алгоритм построения родительского графа над множеством Stereoclique работает за не более чем

$$2|W|^2 + 3|U| + |N|^2 + 3|N| + |S| + 2|U|\log_2|U| + \\ + \sum_{u\in N} |D_u| + 2|V_u| \cdot \log_2|V_u| + \sum_{u\in U} 2e_u(\log_2|U| + |V_u|) + \\ + O\left(\sum_{u\in N}\left(|S_u|^2 + \log|S_u| \cdot \sum_{v\in D_u}|P_v|\right)\right),$$

где W — набор МФЗ, U — множество значимых весов, e_u — число пар вершин, пересечение весов которых в точности равно u; V_u — множество вершин, входящих в значимую клику веса u, N — множество NotBiclique, S — множество Stereoclique, S_u — множество сыновей значимой клики веса u, D_u — множество потомков значимой клики веса u, P_u — множество родителей значимой клики веса u.

Доказательство. Это непосредственно следует из утверждений 4 и 5.

Замечание 3. Поскольку алгоритмы построения родительского графа над множеством StereoClique, приведенные в листингах 5 и 6, различаются так же, как алгоритмы построения пустого графа над множеством StereoClique, приведенные в листингах 7 и 8, то и время работы у них соотносится так же.

10. Заключение. В работе рассмотрены 6 алгоритмов построения элементов третичной полиструктуры АБС, опубликованные ранее как части алгоритмов построения множества минимальных графов смежности или единственного произвольного представителя этого множества [27–31, 35].

Из этих алгоритмов четыре относятся к построению пустого графа над некими подмножествами сужений, а еще два строят родительский граф над множеством стереоклик.

Алгоритм построения пустого графа смежности над неким множеством совпадает с алгоритмом построения этого множества. При этом алгоритм построения множества Clique работает заведомо быстрее алгоритма построения NotBiclique, который, в свою очередь, работает заведомо быстрее алгоритма построения Steroclique (при этом $\text{Steroclique} \subseteq \text{NotBiclique} \subseteq \text{Clique}$), потому что любой алгоритм из рассмотренных строит подмножества путем исключения элементов множества.

Скорость работы двух алгоритмов, строящих множества Stereoclique, в статье была сравнена, однако выявить лучший по этому показателю алгоритм не удалось: предложены примеры, которые это демонстрируют. Тем не менее, автор склоняется к предположению о том, что алгоритм построения пустого графа над множеством Stereoclique полным перебором работает в среднем быстрее алгоритма построения пустого графа над множеством Stereoclique перебором значимых ребер, то есть предположение, выдвинутое в статье [30] по ускорению работы алгоритма построения множества минимальных графов смежности себя, вероятно, не оправдало. Однако следует отметить, что алгоритм построения пустого графа над множеством Stereoclique (точно так же, как и соответствующий алгоритм построения множества минимальных графов смежности при помощи клик владений [30]) строит дополнительно четвертичную структуру АБС, которая представляет самостоятельный интерес для исследования [36].

Различие между двумя рассмотренным алгоритмами построения родительского графа над множеством Stereoclique сводится исключи-

тельно к разнице между построением указанного множества (указанные алгоритмы полностью включают в себя соответствующие алгоритмы построения пустого графа над множеством Stereoclique). С точки зрения построения родительского графа, эти алгоритмы повторяют алгоритм построения родительского графа снизу–вверх [32].

Остаются открытыми вопросы о том, какой из алгоритмов построения множества Stereoclique быстрее в среднем и быстрее асимптотически, потому что приведенная оценка зависит от многих параметров рассматриваемой структуры и в явном виде не может быть упрощена (требуются дополнительные исследования о соотношениях и взаимных ограничениях используемых параметров структуры АБС). Также остается открытым такой же вопрос и для алгоритмов построения родительских графов над множеством Stereoclique, потому что разница в работе различающихся частей построения основного множества вершин значимой клики может оказаться несущественной по отношению к общему времени работы алгоритмов построения родительского графа над множеством Stereoclique.

Наконец, уточнение оценки работы алгоритмов также будет серьезным достижением. По приведенной оценке можно говорить о полиномиальной сложности алгоритмов. Грубая оценка позволит говорить о четвертой степени для всех алгоритмов, однако вероятно, что при уточнении возможных соотношений между исследуемыми параметрами степень может быть понижена.

Следует также отметить, что в работе были рассмотрены алгоритмы построения четырех элементов третичной полиструктуры. Сама полиструктура по определению состоит из значительного числа элементов: из всех возможных направленных графов, построенных над всеми возможными подмножествами сужений. Перечисление всех этих элементы и, тем более, их построение неактуальны, поскольку интерес представляют только действительно используемые элементы третичной полиструктуры, алгоритмы построения которых и приведены в настоящей работе.

Литература

1. *Городецкий В.И.* Алгебраические байесовские сети — новая парадигма экспертных систем // Юбилейный сборник трудов институтов Отделения информатики, вычислительной техники и автоматизации РАН. Т. 2. М.: РАН. 1993. С. 120–141.
2. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы. Построение и анализ. 2-е изд. М.: Вильямс. 2005. 1296 с.

3. *Опарин В.В., Тулупьев А.Л.* Синтез графа смежности с минимальным числом ребер: формализация алгоритма и анализ его корректности // Труды СПИИРАН. СПб.: Наука, 2009. Вып. 11. С. 142–157.
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. *Сироткин А.В., Тулупьев А.Л.* Моделирование знаний и рассуждений в условиях неопределенности: матрично-векторная формализация локального синтеза согласованных оценок истинности // Труды СПИИРАН. 2011. Вып. 18. [в печати].
7. *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).
8. *Тулупьев А.Л.* Алгебраические байесовские сети. Логико-вероятностный подход к моделированию баз знаний с неопределенностью. СПб.: СПИИРАН, 2000. 292 с.
9. *Тулупьев А.Л.* Алгебраические байесовские сети: локальный логико-вероятностный вывод: Учеб. пособие. СПб.: СПбГУ; ООО Издательство «Анатолия», 2007. 80 с. (Сер. Элементы мягких вычислений).
10. *Тулупьев А.Л.* Алгебраические байесовские сети: логико-вероятностные графические модели баз фрагментов знаний с неопределенностью: Диссертация на соискание ученой степени д-ра физ.-мат. наук. СПб., 2009. 670 с. (Санкт-Петербургский государственный университет).
11. *Тулупьев А.Л.* Алгебраические байесовские сети: система операций локального логико-вероятностного вывода // Информационно-измерительные и управляющие системы. 2009. №4. С. 41–44.
12. *Тулупьев А.Л.* Алгебраические байесовские сети: система операций глобального логико-вероятностного вывода // Информационно-измерительные и управляющие системы. 2010. №11. С. 65–72.
13. *Тулупьев А.Л.* Алгебраические байесовские сети. Теоретические основы и непротиворечивость. СПб.: СПИИРАН, 1995. 76 с.
14. *Тулупьев А.Л.* Апостериорные оценки вероятностей в идеале конъюнктов // Вестник СПбГУ. 2010. Серия 10. Вып. 1. С. 95–104.
15. *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.
16. *Тулупьев А.Л.* Байесовские сети: логико-вероятностный вывод в циклах. СПб.: Изд-во С.-Петербургского ун-та, 2008. 140 с. (Элементы мягких вычислений).
17. *Тулупьев А.Л.* Генерация множества ограничений на распределение оценок вероятности над идеалом цепочек конъюнкций // Вестник молодых учёных. 2004. № 4. Серия: Прикладная математика и механика. 2004. № 1. С. 35--43.
18. Тулупьев А.Л. Метод построения и исследования баз фрагментов знаний с неопределенностью // Труды СПИИРАН. 2002. Вып. 1, т. 1. СПб.: Наука, 2002. С. 258—271.
19. *Тулупьев А.Л.* Непротиворечивость оценок вероятностей в алгебраических байесовских сетях. Вестник СПбГУ. Сер. 10. 2009. Вып. 3. С. 144–151.

20. *Тулупьев А.Л.* Непротиворечивость оценок вероятностей в идеалах конъюнктов и дизъюнктов. Вестник СПбГУ. Сер. 10. 2009. Вып. 2. С. 121–131.
21. *Тулупьев А.Л.* Основы теории алгебраических байесовских сетей: программа спецкурса для студентов старших курсов и аспирантов. СПб.: СПбГУ, 2007. 7 с.
22. *Тулупьев А.Л.* Оценка чувствительности результата априорного логико-вероятностного вывода в интеллектуальных информационных системах // Известия высших учебных заведений: Приборостроение. 2009. №9. С. 3–6.
23. *Тулупьев А.Л.* Согласованность данных и оценка вероятности альтернатив в цикле стохастических предпочтений // Известия высших учебных заведений: Приборостроение. 2009. № 7. С. 3–8.
24. *Тулупьев А.Л., Николенко С.И., Сироткин А.В.* Байесовские сети: логико-вероятностный подход. СПб.: Наука, 2006. 607 с.
25. *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.
26. *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.
27. *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.
28. *Фильченков А.А.* Алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик // Труды СПИИРАН. 2010. Вып. 1 (12). С. 119–133.
29. *Фильченков А.А.* Алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик-собственников // Труды СПИИРАН. 2010. Вып. 3 (14) С. 150–169.
30. *Фильченков А.А.* Алгоритм построения множества минимальных графов смежности при помощи клик владений // Труды СПИИРАН. 2010. Вып. 2 (13). С. 119–133.
31. *Фильченков А.А.* Алгоритм построения множества минимальных графов смежности при помощи клик-собственников владений // Труды СПИИРАН. 2010. Вып. 4 (15). С. 193–212.
32. *Фильченков А.А.* Алгоритмы построения третичной структуры алгебраической байесовской сети // Труды СПИИРАН. 2011. Вып. 17. С. 197–218.
33. *Фильченков А.А., Тулупьев А.Л.* Анализ циклов в минимальных графах смежности алгебраических байесовских сетей // Труды СПИИРАН. 2011. Вып. 17. С. 151–173.
34. *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.
35. *Фильченков А.А., Тулупьев А.Л.* Структурный анализ систем минимальных графов смежности Труды СПИИРАН. 2009. Вып. 11. С. 104–127.
36. *Фильченков А.А., Тулупьев А.Л.* Третичная структура алгебраической байесовской сети // Труды СПИИРАН. 2011. Вып. 18. [в печати].
37. *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.
38. *Фильченков А.А., Тулупьев А.Л., Сироткин А.В.* Мощность множества минимальных графов смежности // Труды СПИИРАН. 2010. Вып. 4 (15). С. 136–161.

39. *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.
40. *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.
41. *Фильченков А.А., Тулупьев А.Л., Сироткин А.В.* Структурный анализ клик минимальных графов смежности // Вестн. Тверск. гос. ун-та. Сер.: Прикладная математика. 2011. №20. С. 139–151.
42. *Харари Ф.* Теория графов. М.: УРСС. 2003, 296 с.
43. *Korb K.B., Nicholson A.E.* Bayesian Artifical Intelligence. NY.: Chapman and Hall/CRC. 2004. 364 p.

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 — 28. 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 — 28. 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 02.08.2011.

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 элементов третичной полиструктуры алгебраической байесовской сети.**

Алгебраические байесовские сети (АБС) представляют собой логико-вероятностную графическую модель баз фрагментов знаний с неопределенностью. При этом под фрагментом знаний понимается идеал конъюнктов над заданным набором атомарных пропозициональных формул с заданным на них вероятностным распределением или семейством распределений. Первичная структура АБС представляет собой набор фрагментов знаний, а вторичная — граф, построенный над этими фрагментами. Выделяют также третичную полиструктуру АБС, которая представляет собой семейство графов, построенных над подмножествам сужений максимального графа смежности.

Интерес к третичной полиструктуре АБС стал возрастать: если изначально она рассматривались лишь как дополнительный объект, возникающий при построении вторичной структуры, то в дальнейшем она стала основной для поиска циклов во вторичной структуре без ее непосредственного построения, а сейчас также обсуждается возможность проводить логико-вероятностный вывод именно на третичной полиструктуре.

Все существующие алгоритмы построения множества минимальных графов смежности или произвольного минимального графа смежности требуют предварительного построения элемента третичной полиструктуры. В работе рассмотрено шесть алгоритмов построения четырех различных элементов третичной полиструктуры.

Так, в частности, рассмотрены алгоритмы построения пустого графа над множеством значимых клик, над множеством значимых клик за исключением множества биклик и два алгоритма — над множеством стереоклик, а также два алгоритма построения родительского графа над множеством стереоклик. Для всех алгоритмов предложена оценка времени их работы.

Алгоритм построения пустого графа над множеством значимых клик работает быстрее, чем алгоритм построения пустого графа над множеством значимых клик за исключением биклик, который работает быстрее, чем алгоритм построения пустого графа над множеством стереоклик. Два алгоритма построения пустого графа над множеством стереоклик работают для разных первичных структур с различным результатом, оценка зависит от множества характеристик первичной структуры. Алгоритмы построения родительского графа опираются на алгоритмы построения пустого графа. Оба алгоритма различаются так же, как и соответствующие алгоритмы построения пустого графа над множеством стереоклик.

ABSTRACT

Filchenkov A.A. **Algebraic Bayesian Network Tertiary Polystructure Elements Synthesis Algorithms.**

Algebraic Bayesian networks (ABN) are logical and probabilistic graphical models of bases of knowledge pattern with uncertainty. A knowledge pattern is an ideal of conjuncts of a given set of atomic propositional formulas with a given probability distribution on them or a family of distributions. The ABN primary structure is a collection of knowledge patterns and the ABN secondary structure is a graph built over these knowledge patterns. The ABN tertiary polystructure, that is a family of graphs built over a subset of the maximal join graph narrowings, is also marked.

Interest in the ABN tertiary polystructure began to grow presently. In original, it was considered only as an additional object that was needed only for a secondary structure synthesis. Later it became the basis for finding a secondary structure cyclicity without its direct synthesis. Now the possibility to carry out logical and probabilistic inference my means of the tertiary polystructure is being considered.

All the existing algorithms for synthesizing the minimal join graph set or a random single minimal join graph require a tertiary polystructure element already synthesized. Six different algorithms for synthesizing four different elements of the tertiary polystructure are overviewed in the paper.

Thus, in particular, the algorithms for synthesizing the empty graph over the useful clique set, over useful clique set except bicliques and (two algorithms) over the stereoclique set are presented in the paper, as well as two algorithms for synthesizing the parent graph over the stereoclique set. Complicity estimations are proposed for all the mentioned algorithms.

The algorithm for synthesizing the empty graph over the useful clique set runs faster than the algorithm for synthesizing the empty graph over the useful clique set bicliques. The last algorithm runs faster than the algorithm for synthesizing the empty graph over the stereoclique set. Two algorithms for synthesizing the empty graph over the stereoclique set run with different results for different primary structures. The estimation depends on many primary structure properties. The algorithms for synthesizing the parent graph are based on the empty graph synthesis algorithms. Both algorithms for synthesizing the parent graph differ in the same way as the corresponding algorithms for synthesizing the empty graph over the stereoclique set.