Санкт-Петербургский государственный университет
Кафедра информатики
Фундаментальная информатика и информационные технологии

Левенец Даниил Григорьевич

АЛГЕБРАИЧЕСКИЕ БАЙЕСОВСКИЕ СЕТИ: СИСТЕМА АНАЛИЗА И СИНТЕЗА ВТОРИЧНОЙ СТРУКТУРЫ

Бакалаврская работа

Научный руководитель:
проф. каф. инф., д.ф.-м.н., доц. Тулупьев А.Л.

Рецензент:
доцент НИУ ВШЭ СПб., к.ф.-м.н. Сироткин А.В.

Санкт-Петербург
2016

Saint-Petersburg State University
Computer Science Department
Fundamental Informatics and Information Technology

Daniel Levenets

ALGEBRAIC BAYESIAN NETWORKS: SYSTEM FOR ANALYSIS AND SYNTHESIS OF SECONDARY STRUCTURE

Bachelor's Thesis

Scientific supervisor:
doctor. of Sc., associate professor Alexander Tulupyev

Reviewer:
associate professor NRU HSE, PhD Alexander Sirotkin

Saint-Petersburg
2016

Содержание

Введение

Графы смежности интенсивно используются в алгебраических байесовских сетях (далее АБС) в качестве вторичной структуры [9, 11, 16, 30, 33]. Такой объект оказался необходим потому, что ряд алгоритмов локального и глобального вывода, использующие рекуррентный подход, требуют, чтобы вторичной структурой был ациклический минимальный граф смежности [1, 8, 10, 12–14]. Однако предложенная стуктура сложна в построении, что заметно усложняет работу с динамически изменяющимися данными. Инкрементальный, а также использующий подобные методы декрементальный алгоритмы, рассматриваемые в данной работе, применяются для перестроения вторичной структуры АБС, что позволяет динамически изменять структуру графа смежности, не прибегая к полному перестроению графа. Это существенно сказывается на времени работы подобных алгоритмов вывода [2,5]. Кроме того, минимальный граф смежности легче визуализировать, равно как и легче выявлять скрытые закономерности, им представленные.

При сложившейся практике обмена данными [15, 32] между программным компонентом, реализующим графический пользовательский интерфейс, и «блоком вычислений» — программным компонентом, обеспечивающим построение сложной системы из заданных составляющих, — типичный сценарий работы предполагает формирование набора составляющих, отправку набора в блок вычислений, формирование сложной системы, отправку данных в графический пользовательский интерфейс для визуального представления сформированной системы. Особенно это справедливо для клиент-серверных систем с так называемым «тонким» клиентом.

Однако этот типичный сценарий ведёт ко всё более длительному ожиданию оператора, когда набор составляющих растёт, поскольку зачастую вычислительная сложность алгоритмов синтеза экспоненциально зависит

от объема исходных данных или оказывается ещё большей.

Одним из возможных путей решения или, по крайней мере, смягчения обозначенной проблемы является инкрементализация и дектрементализация алгоритмов, которая нацелена на использование того, что было построено на предшествующих шагах работы с программой.

Иначе говоря, на практике при работе с большим набором данных, в котором имеются связи, зависимости и отношения, часто ставится задача изменения подобного набора данных таким образом, чтобы имеющиеся ранее связи, зависимости и отношения либо сохранялись прежними, либо время, затрачиваемое на их перестроение, было бы близким к минимальному [24, 31, 34, 35, 37]. Трудозатраты на выполнение таких операций играют важную, а порой и критическую роль, когда речь идет о real-time-системах или о системах, для которых важна высокая степень доступности. Таким образом, возникает вопрос, как избежать полного перестроения системы после каждого небольшого изменения исходного набора данных.

В рассматриваемом случае — примере минимального графа смежности как сложной системы — осуществляется попытка использовать уже построенный минимальный граф смежности и либо дополнить его новой вершиной, либо обеднить одной из существующих вершин с соблюдением требований структуры, подробно описанных во 2-м разделе.

Таким образом, теоретической целью данной работы является разработка инкрементальных и декрементальных алгоритмов, ускоряющих построение глобальных структур алгебраических байесовских сетей при работе с динамически изменяющимися данными.

Таже в данной работе есть технологическая цель: создание системы анализа и синтеза вторичной структуры алгебраических байесовских сетей в виде комплекса программ, который позволял бы пользователям наглядно изучать поведение глобальных структур АБС при поступлении в сеть новых данных, а также проводить различного рода вычисления, важные

в контексте вероятностных графических моделей. Такая система могла бы пригодиться при изучении и преподавании теории алгебраических байесовских сетей. Пользователи системы получат возможность быстро и удобно работать с такой сложной структурой, как граф смежности, и для этого им не понадобится изучать особенности программной реализации подобных алгебраических структур. Также необходимо отметить, что настоящая выпускная квалификационная работа бакалавра является проектной, то есть выполняется совместно с коллегами (Романовым Артемом Витальевичем, Березиным Алексеем Ивановичем), кроме того, тесно связана с выпускной квалификационной работой бакалавра Мальчевской Екатерины Андреевны.

Для удобства изложения сначала раскрывается теоретическая часть данной работы (главы 3–5), а затем практическая (глава 6), при этом все используемые определения вынесены в главу 2.

1 Обзор существующих алгоритмов синтеза вторичной структуры АБС

В данной главе проводится обзор существующих решений, касающихся синтеза вторичной структуры АБС. Обосновывается актуальность выбранной для работы темы.

1.1 Существующие решения

В работах В.В.Опарина и А.А.Фильченкова были предложены два классических алгоритма синтеза вторичной структуры АБС — прямой и жадный [6, 7, 20, 22]. Оба алгоритма генерируют вторичную структкуру по полному набору данных, однако обладают серьезным недостатком — при малейшем изменении входного набора данных они полностью перестраивают

сгенерированную ранее структуру, то есть не учитывают текущее состояние сети и строят ее с нуля [2, 5]. Такой подход может быть полезен при синтезе начального состояния вторичной структуры, однако для дальнейших ее изменений не подходит.

1.2 Актуальность работы

Обратимся к более детальному рассмотрению существующего положения вещей и средств его улучшения.

Для того, чтобы граф являлся графом смежности, необходимо поддерживать выполнение определённых условий (см. определение из раздела 2), результат проверки которых зависит от текущего набора данных; также, чтобы граф смежности являлся *минимальным* графом смежности, необходимо поддерживать выполнение других условий, которые, в свою очередь, также зависят от текущего набора данных.

В отношении предложенных ранее жадного и прямого алгоритмов синтеза минимального графа смежности были построены теоретические оценки сложности, а также проведён статистический анализ сложности этих двух конкурирующих алгоритмов [3,4,29]. Результаты статистического анализа отношений скорости работы двух алгоритмов позволили выделить три поддиапазона мощности наборов вершин графов смежности: в поддиапазоне 5–35 жадный алгоритм работает существенно быстрее прямого, в поддиапазоне 60–105 прямой алгоритм работает существенно быстрее жадного алгоритма, а в поддиапазоне 35–60 выигрыш в скорости зависит от конкретного набора исходных данных. Кроме того, было установлено, что в диапазоне 5–60 имеется некоторое число статистических выбросов, сигнализирующих об особенностях в соответствующих наборах исходных данных. Наконец, следует учесть, что на достаточно большом множестве наборов нагрузок, а именно, когда мощность набора превышает некоторый уровень, оба алгоритма работают долго, в частности, и с субъективной

точки зрения пользователя тоже.

Именно поэтому особенно актуальной задачей представляется добавление новой вершины v в уже в существующий минимальный граф смежности $G = \langle V, E \rangle$ таким образом, чтобы граф $G' = \langle V \cup \{v\}, E' \rangle$ также оказался бы минимальным графом смежности, а время, затраченное на построение множества E', было бы минимальным, либо хотя бы приемлемым с точки зрения пользователя или решаемых задач машинного обучения.

2 Определения и обозначения

В данной главе описываются основные понятия и определения из теории алгебраических байесовских сетей, используемые в дальнейшем в работе, а также технологии, применяемые при разработке комплекса программ.

2.1 Теоретическая часть

Воспользуемся системой терминов и обозначений, сложившихся в ранее опубликованных работах по данной тематике [6, 9, 13, 14, 16]. Пусть задан конечный алфавит символов , а непустые множества символов (без повторов) — слова — рассматриваются как возможные значения нагрузок вершин графов (в основном, графов смежности) и их ребер. В контексте настоящей работы вершины и их нагрузки соотносятся взаимооднозначно, поэтому мы будем использовать соответствующие им обозначения v и W_v взаимозаменяемо.

Определение 1. Нагрузка вершины является *допустимой*, если она не совпадает с нагрузкой никакой другой вершины, не поглощается нагрузками и не поглощает нагрузки никаких других вершин.

Определение 2. Назовем неориентированный граф $G = \langle V, E \rangle$ графом смежности, если он удовлетворяет следующим условиям:

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

$$W_u \cap W_v \subseteq W_s;$$

2. $\forall \{u, v\} \in E \;\; W_u \cap W_v \neq \emptyset$.

При этом множество вершин такого графа будем называть его *первичной структурой*.

Граф смежности с минимальным и максимальным числом ребер мы будем называть *минимальным графом смежности* и *максимальным графом смежности* соответственно.

Определение 3. Пересечение нагрузок двух вершин $W_u \cap W_v$ будем называть *сепаратором*. Элементами сепаратора являются нагрузки.

Введём ряд дополнительных обозначений, которые понадобятся нам при описании алгоритмов.

PathExists(G, edge, nodeBegin, nodeEnd) — функция, которая определяет, связаны ли магистрально вершины nodeBegin и nodeEnd графе $G' = \langle V, E' \rangle$, причем, $G = \langle V, E \rangle$, и $E' = \{\{p, q\} : \{p, q\} \in E \;\&\; \{p, q\} \neq \text{edge}\}$.

edge.First, edge.Second — вершины ребра edge.

edge.RemoveAllowed — метка, которая показывает, возможно ли удаление ребра edge или нет. Изначальное значение — **TRUE**.

Пусть задан конечный алфавит символов A, а непустые множества символов (без повторов) — слова — рассматриваются как возможные значения нагрузок вершин графов (в основном, графов смежности) и их рёбер. Пусть имеется набор вершин $V = \{v_1, v_2, \ldots, v_n\}$ и такие нагрузки $W = \{W_1, W_2, \ldots, W_n\}$, причем $\forall u \in V \;\; W_u$ является нагрузкой для вершины u.

2.2 Технологическая часть

Windows Presentation Foundation (WPF) [40] — система для построения клиентских приложений с визуально привлекательными возможностями взаимодействия с пользователем [43], графическая (презентационная) подсистема в составе .NET Framework [45] (начиная с версии 3.0), использующая расширяемый язык разметки XAML [44].

Model-View-ViewModel (MVVM) [39] — шаблон проектирования архитектуры приложения [42], ориентированный на современные платформы разработки, такие как Windows Presentation Foundation, Silverlight [38] от компании Microsoft [27].

Шаблон MVVM делится на три части:

- Модель (англ. Model), так же, как в классической MVC (Model-View-Controller) — представляет собой фундаментальные данные, необходимые для работы приложения.

- Представление (англ. View) — это графический интерфейс, то есть окно, кнопки и так далее. Представление является подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью представления. В случае, если в Модели представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновленное значение свойства из Модели представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью представления.

- Модель представления (англ. ViewModel) — является, с одной стороны, абстракцией Представления, а с другой, предоставляет обёртку данных из Модели, которые подлежат связыванию. То есть она содержит Модель, которая преобразована к Представлению, а также содержит

в себе команды, которыми может пользоваться Представление, чтобы влиять на Модель.

3 Инкрементальный алгоритм добавления вершины в минимальный граф смежности

Задан минимальный граф смежности G, на вход алгоритму поступает указанный граф и новая вершина. Требуется построить новый минимальный граф смежности — такой, что множество вершин состоит из объединения множества вершин V графа G и новой вершины. При этом для простоты изложения, предполагаем, что набор нагрузок остается корректным: нагрузка новой вершины допустима, то есть её нагрузка и нагрузка любой вершины из множества V не совпадают, а также нагрузка новой вершины не поглощает полностью нагрузку любой вершины из множества V и не поглощается такими нагрузками. Используя формальные обозначения, то же самое запишем следующим образом.

$G = \langle V, E \rangle$ — граф, подаваемый на вход алгоритму. v — новая вершина, подаваемая на вход алгоритму. $G' = \langle V \cup \{v\}, E' \rangle$, где E' — множество ребер минимального графа смежности G'. Как отмечалось ранее, инкрементальный алгоритм берёт за основу построения минимального графа смежности G' множество рёбер E — благодаря такому подходу предполагается, что инкрементальный алгоритм будет работать быстрее жадного (прямого) алгоритма, так как большая или даже большая часть связей (рёбер) останется без изменений.

Напомним, что в контексте данной работы обозначения вершины и её нагрузки совпадают. Такой подход будет удобен для чтения и понимания графиков, представленных ниже, а также в разделе о корректности алгоритма.

3.1 Описание алгоритма

На листинге 1 приведен псевдокод инкрементального алгоритма добавления новой вершины в минимальный граф смежности.

Приведённый алгоритм добавляет новую вершину в существующий минимальный граф смежности G, а на выходе выдаёт минимальный граф смежности G' с добавленной новой вершиной.

Представленный алгоритм состоит из двух частей. В первой части алгоритма (строки 2–7) во вторичную структуру алгебраической байесовской сети добавляются все допустимые рёбра. Во второй (строки 8–19) части алгоритма происходит удаление тех ребер, отсутствие которых не приведёт к нарушению магистральной связности вершин графа, другими словами — синтезируется минимальный граф смежности. Рассмотрим алгоритм более подробно.

В строке 3 в множество вершин V', состоящее в точности из вершин V исходного графа G, добавляется новая вершина v. В строке 4 для каждой вершины из множества V и новой вершины v проверяется наличие непустого сепаратора. Если такой сепаратор есть, то в множество E' добавляется соответствующее ребро.

После добавления новой вершины и новых ребер формируется граф G'. Затем из него необходимо удалить те ребра, отсутствие которых не приведёт к нарушению магистральных связей (строки 8–19). В результате такого преобразования граф G' станет минимальным графом смежности. Отметим, что ребра удаляются с помощью жадного алгоритма, описанного в статьях [4, 6, 7].

Приведённый алгоритм позволяет добавлять новую вершину v в уже существующий минимальный граф смежности G. Отметим, что в указанном алгоритме для добавления новой вершины не требуется заново синтезировать множество рёбер искомого графа смежности, так как инкрементальный алгоритм учитывает и дополняет полученные ранее результаты.

Algorithm 1 Инкрементальный алгоритм добавления новой вершины в минимальный граф смежности.

input: $G = \langle V, E \rangle$, v
output: $G' = \langle V', E' \rangle$

```
1:  function SimpleIncremental
2:      E' = E
3:      V' = V ∪ {v}

4:      foreach (u in V)
5:          if ((W_u ∩ W_v) ≠ ∅) then
6:              E' = E' ∪ {u, v}

7:      G' = ⟨V', E'⟩

8:      while (TRUE)
9:          edge = ∅

10:         foreach(e in E')
11:             if (e.RemoveAllowed) then
12:                 edge = e
13:                 break foreach //Выход из foreach

14:         if (edge == ∅) then
15:             break while //Выход из while

16:         if (PathExists(G', edge, edge.First, edge.Second)) then
17:             E' = E' \ {edge}
18:         else
19:             edge.RemoveAllowed = FALSE

20:     return G'
```

Отметим, что в инкрементальном алгоритме были использованы результаты и объекты, построенные на предыдущих шагах. В данном случае это — минимальный граф смежности, который предстоит дополнить новой вершиной, то есть, собственно основной результат предшествующего запуска алгоритма синтеза. В других случаях могут оказаться доступны и вспомогательные конструкции или объекты, которые использовались при построении основного результата. Например, такое можно ожидать в случае, когда стоит задача построить не один минимальный граф смежности, а все возможные минимальные графы смежности [17–21, 23].

Кроме того, учтём, что потребовалось строить не весь набор ребер максимального графа смежности, а лишь те рёбра, которые исходят из новой вершины. При этом экономия в числе операций может возникнуть, и, исключая особые, специально подобранные случаи, как правило, возникает, за счёт двух моментов. Во-первых, требуется построить меньшее число ребер. Во-вторых, последующий перебор для исключения избыточных рёбер идёт по меньшему их числу.

Наконец, экономия может проявится и за счёт того, что из пользовательского интерфейса пересылаются сведения об одной новой вершине, а не о всей их совокупности. Конечно, в случае нагрузок графа смежности в той форме, в которой они описаны выше, эффект экономии не будет так заметен. Но если вершина в свою очередь определяется сложным объемным набором данных, оператор почувствует ускорение в обработке его команды.

3.2 Корректность алгоритма

3.2.1 Алгоритм синтезирует граф смежности

При добавлении сразу всех допустимых ребрер, исходящих из новой вершины v, в граф смежности, получается новый граф смежности; следовательно, после выполнения первой части алгоритма (строки 2–6), граф

G' также будет графом смежности, при этом допустимость этих рёбер обуславливается проверкой, осуществляемой в строке 5. Проанализируем корректность этой части алгоритма более подробно.

Теорема 1. Пусть $G = \langle V, E \rangle$ — минимальный граф смежности, v — добавляемая (новая) вершина, $V' = V \cup \{v\}$, а $E' = E \cup \{\{v, w\} : w \in V \ \& \ W_v \cap W_w \neq \emptyset\}$, тогда $G' = \langle V', E' \rangle$ — граф смежности.

Доказательство: Возьмём любые две вершины s и $r \in G'$. Рассмотрим два случая.

Сепаратор s и $r \neq \emptyset$. Тогда докажем, что в графе G' существует путь между заданными вершинами. Предположим, что обе данные вершины принадлежат исходному графу (s и $r \in G$), тогда, по определению графа смежности, между данными вершинами существует некоторый путь. Если же одна из данных вершин не принадлежит исходному графу, то она совпадает с вершиной v ($s = v$ или $r = v$), следовательно, по определению множества $E' = E \cup \{\{v, w\} : w \in V \ \& \ W_v \cap W_w \neq \emptyset\}$ — между этими вершинами существует ребро, которое и является путём с нужными свойствами, то есть магистралью.

Сепаратор s и r равен $\emptyset$. Тогда докажем, что в графе G' не существует магистрали между этими вершинами. Предположим, что обе вершины принадлежат исходному графу(s и $r \in G$), тогда, по определению графа смежности, между данными вершинами не существует ребра. Если же одна из вершин не принадлежит исходному графу, то она совпадает с вершиной v ($s = v$ или $r = v$), следовательно, по определению множества $E' = E \cup \{\{v, w\} : w \in V \ \& \ W_v \cap W_w \neq \emptyset\}$ — между данными вершинами не существует ребра.

Таким образом, любая пара вершин, пересечение нагрузок которых непусто, соединены магистралью, а вершины, нагрузки которых не пересекаются, не соединены ребром. Граф, обладающий таким свойством, является графом смежности. Также следует отметить, что полученный на пер-

вом шаге граф не обязательно является минимальным графом смежности. Приведём пример.

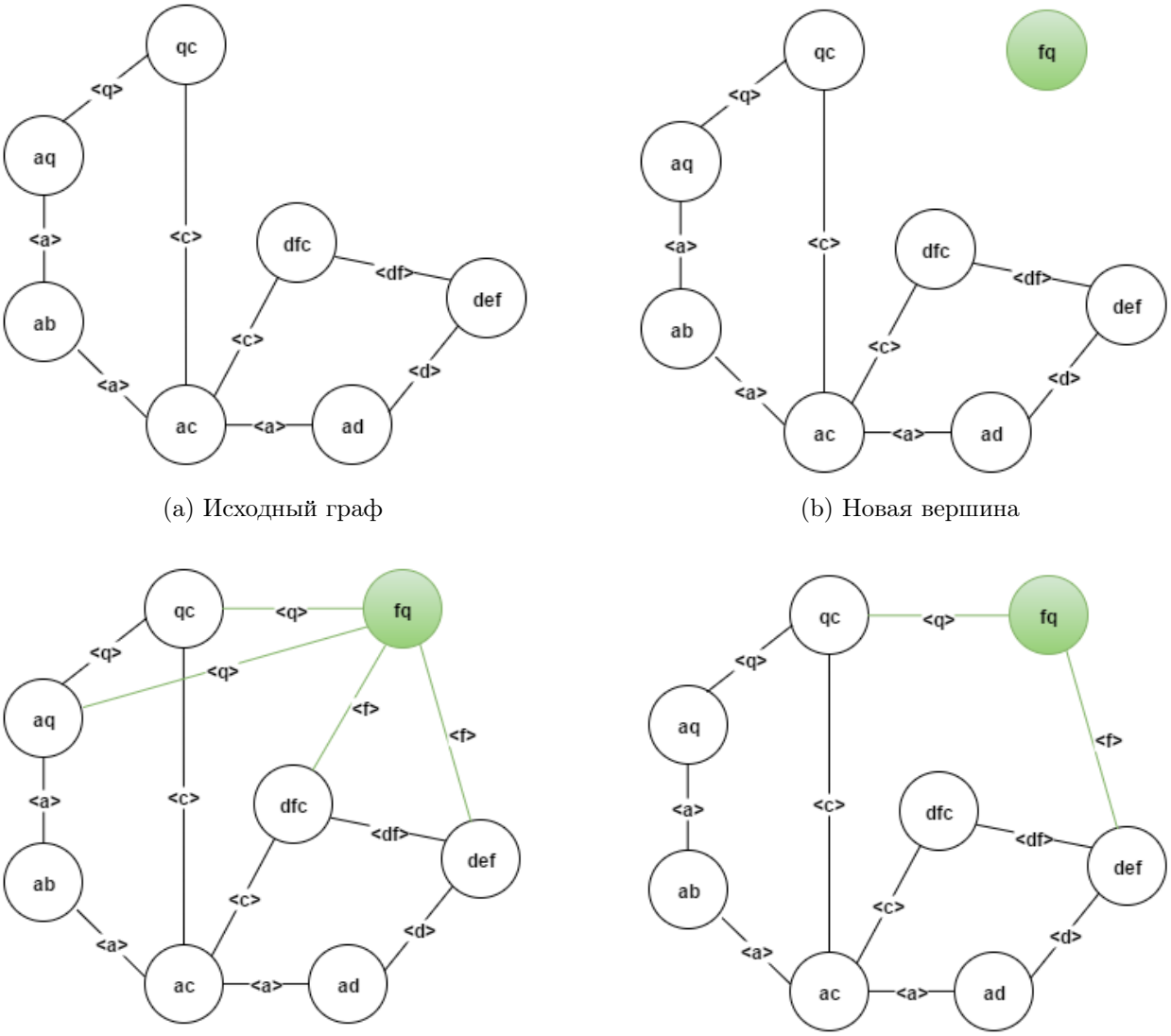
В минимальный граф смежности G (рис. 1a) добавляется новая вершина v с нагрузкой $\{f, q\}$ (рис. 1b). На первом шаге инкрементального алгоритма в граф добавляются несколько рёбер с нагрузками $\{q\}$ и $\{f\}$ (рис. 1c). Заметим, что полученный граф G' не является минимальным графом смежности, так как удаление ребра $\{aq, fq\}$ или любого другого ребра с нагрузкой $\{q\}$ не приводит к нарушению свойств графа смежности (рис. 1d).

3.2.2 Алгоритм синтезирует минимальный граф смежности

Теорема 2. Алгоритм (листинг 1) синтезирует минимальный граф смежности.

Доказательство. Удаление ребра при сохранении магистральности есть не что иное, как переход от одного графа смежности к другому с уменьшением числа ребер. В статьях [6, 7] было доказано, что множество графов смежности формирует матроид.

По свойствам матроида, последовательность допустимых удалений рёбер в графе смежности сходится, а полученный в результате завершившейся последовательности допустимых удалений граф окажется минимальным графом смежности. Действительно, как было показано ранее, после выполнения первой части алгоритма, граф G' является графом смежности, а во второй части алгоритма происходят удаления допустимых рёбер. Таким образом, по свойствам матроида, алгоритм из листинга 1 на выходе формирует минимальный граф смежности, включающий новую вершину v, что и требовалось доказать.



(a) Исходный граф

(b) Новая вершина

(c) Новые ребра

(d) Новый минимальный граф смежности

Рис. 1: Интеграция новой вершины в граф смежности с сохранением его минимальности.

4 Улучшенный инкрементальный алгоритм добавления вершины в минимальный граф смежности

Формальная постановка задачи эквивалентна задаче из раздела 3. Задан минимальный граф смежности G, на вход алгоритму поступает указанный граф и новая вершина. Требуется построить новый минимальный граф смежности — такой, что множество его вершин состоит из объединения множества вершин V графа G и новой вершины. При этом для простоты изложения предполагаем, что набор нагрузок остается корректным: нагрузка новой вершины допустима, то есть её нагрузка и нагрузка любой вершины из множества вершин V не совпадают, а также нагрузка новой вершины не поглощает полностью нагрузку любой вершины из множества V и не поглощается такими нагрузками.

Однако по сравнению с предыдущим алгоритмом, приведённый в данном разделе алгоритм дополнительно улучшен тем, что производится отбор кандидатов в новые рёбра: при построении связей из новой вершины берётся не вся возможная совокупность исходящих из неё ребер в вершины, нагрузки которых имеют попарно непустые пересечения с нагрузкой новой, а лишь те из них, сепаратор на которых отвечает определённым условиям.

4.1 Описание алгоритма

На первом шаге пополнения уже построенного минимального графа смежности G новой вершиной v требуется провести в нём новые рёбра, исходящие из v в другие вершины графа, причём, с одной стороны, пополненный граф должен остаться графом смежности, то есть сохранить свойство магистральной связности, а с другой стороны, среди новых рёбер не должно оказаться заведомо избыточных, то есть легко отсеиваемых

по некоторому критерию рёбер. Для построения такого набора рёбер введем функцию GetVerticesToConnect(G, u), программная реализация которой приведена на листинге 2.

Algorithm 2 Алгоритм вычисления множества «необходимых» вершин.

input: $G = \langle V, E \rangle$, v
output: V'

```
1:  function GetVerticesToConnect
2:      S' = ∅
3:      V' = ∅
4:      H =< {V', S} >

5:      foreach (u in V)
6:          sep = W_u ∩ W_v
7:          if (sep == ∅) then
8:              continue foreach
9:              AddAllowed = TRUE

10:         foreach (h in H)
11:             if (sep ⊆ h.S) then
12:                 AddAllowed = FALSE
13:                 breake foreach
14:             if (sep ⊃ h.S) then
15:                 H = H {v, h.S}
16:         if (AddAllowed) then
17:             H = H ∪ {(h.S, sep)}

18:     return H.V'
```

На первом шаге алгоритма вводится дополнительное множество H (строка 4) — множество упорядоченных пар «вершина–сепаратор», которое будет содержать текущий набор «нужных» вершин и их сепараторы с вершиной v. Такая реализация позволяет не вычислять каждый раз сепаратор заново. Необходимо уточнить, что под удалением вершины из множества H подразумевается удаление упорядоченной пары, включающей данную

вершину (строка 15).

Далее, для всех вершин u из множества вершин входного графа (строка 5), сепаратор с вершиной v которых непуст (строка 7), объявляется флаг AddAllowed (строка 9), который указывает на уникальность вершины и соответствующего данной вершине сепаратора. Изначально он принимает значение **TRUE**. После этого вычисленный в строке 6 сепаратор сравнивается со всеми сепараторами $h.S$ из H. При этом возможны 3 случая:

1. Сепаратор является подмножеством сепаратора s или равен ему (строка 11). Это означает, что вершины, имеющие сепаратор s, уже связаны с v, и добавление ещё одного ребра с такой же нагрузкой избыточно (строка 12). Более того, при добавлении этого ребра произойдет образование в графе G цикла с нагрузкой s (см. пример на рис.2).

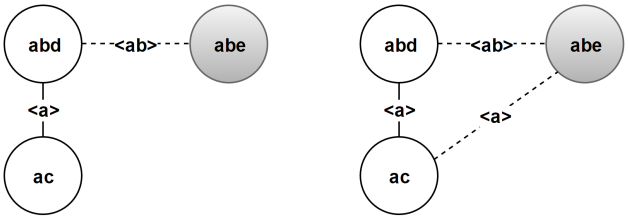


Рис. 2: Цикл с нагрузкой $\langle$ a $\rangle$.

2. s является подможеством сепаратора (строка 14). В этом случае новое ребро покроет собой все вершины, которые покрывались ребром с нагрузкой s и, возможно, ещё некоторые из них. Это означает что, после добавления нового ребра, ребро с нагрузкой s станет избыточно, и его нужно удалить из H (строка 15) (см. пример на рис.3)

3. Все остальные случаи. Тогда сепараторы либо не пересекаются, либо пересекаются частично, а значит, каждый из них покрывает своё уникальное множество вершин и не может быть поглощён другим. Следует отметить, что если пересечение сепараторов не пусто, то в граф может

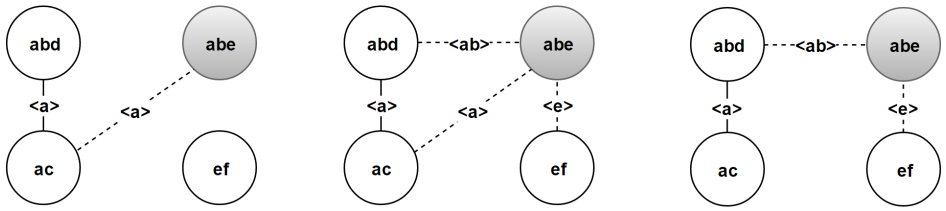


Рис. 3: Ребро ⟨ a ⟩ поглощается ⟨ ab ⟩.

добавиться неэлиминируемый (неустранимый) цикл с нагрузкой, равной пересечению данных сепараторов (если нагрузка ребра не между вершинами G не является подмножеством нового ребра).

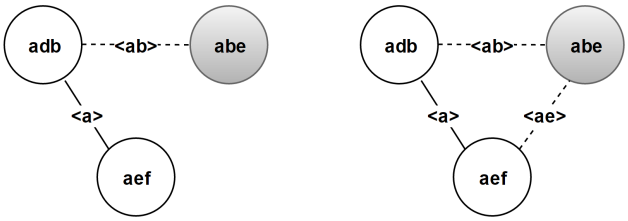


Рис. 4: Элиминируемый цикл с нагрузкой ⟨ a ⟩.

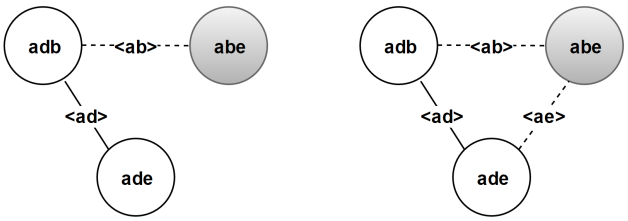


Рис. 5: Неэлиминируемый цикл с нагрузкой ⟨ a ⟩.

Если после описанных выше проверок значение флага AddAllowed осталось истинным, то множество H дополняется новой упорядоченной парой $(u, W_u \cap W_v)$, и, после завершения цикла/обхода всех вершин u графа G, будет состоять из минимального по мощности множества вершин и их сепараторов, добавление ребра к которым необходимо для сохранения свойств магистральности графа G.

Описанные выше результаты использованы в алгоритме на листинге 3. Этот алгоритм инкрементальный, он добавляет новую вершину в минимальный граф смежности. Ему на вход поступают уже существующий

Algorithm 3 Улучшенный инкрементальный алгоритм добавления новой вершины в минимальный граф смежности.

input: $G = \langle V, E \rangle$, v
output: $G' = \langle V', E' \rangle$

```
1: function SmartIncremental
2:     E' = E
3:     V' = V ∪ {v}

4:     foreach (u in GetVerticesToConnect(G, v))
5:         E' = E' ∪ {u, v}

6:     G' =< V', E' >

7:     while (TRUE)
8:         edge = ∅

9:         foreach(e in E')
10:            if (e.RemoveAllowed) then
11:                edge = e
12:                break foreach //Выход из foreach

13:        if (edge == ∅) then
14:            break while //Выход из while

15:        if (PathExists(G', edge, edge.First, edge.Second)) then
16:            E' = E' \ {edge}
17:        else
18:            edge.RemoveAllowed = FALSE

19:    return G'
```

минимальный граф смежности G и новая вершина, а алгоритм в свою очередь возвращает минимальный граф смежности G' с добавленной новой вершиной. Алгоритм улучшен в части предварительного отбора новых рёбер.

Код алгоритма делится на две части. В первой части (строки 2–6) для исходного графа G вычисляется минимальное по мощности множество вершин, проведение рёбер от которых в новую вершину обеспечит магистральность графа G', то есть сделает его графом смежности, и для каждой такой вершины в граф добавляется соответствующее ребро. Во второй части (строки 7–18) из графа удаляются все ребра, отсутствие которых не нарушает магистральную связность графа.

4.2 Корректность алгоритма

4.2.1 Алгоритм синтезирует граф смежности

Теорема 3. Пусть $G = \langle V, E\rangle$ — минимальный граф смежности, v — добавляемая в него вершина, $V' = V \cup \{v\}$, $H = \{h : h \in V, W_h \cap W_v \neq \emptyset, \forall h' \in H : h' \neq h, h \notin h'\}$, а $E' = E \cup \{\{u, w\} : w \in H\}$, тогда $G' = \langle V', E'\rangle$ — граф смежности.

Доказательство. Возьмем любые две вершины $s, r \in G'$. Возможны два случая.

Первый — сепаратор s и $r \neq \emptyset$, тогда докажем, что в G' вершины s и r магистрально связаны. Предположим, что обе вершины принадлежат исходному графу, тогда, по определению графа смежности, между ними существует магистраль. Если же одна из вершин не принадлежит исходному графу, то она совпадает с вершиной v, а значит, по определению множества $E' = E \cup \{\{u, w\} : w \in H\}$ и построению множества H, между ними существует магистраль.

Второй — сепаратор s и $r = \emptyset$, тогда докажем, что в G не существует

ребра между этими вершинами. Предположим, что обе вершины принадлежат исходному графу, тогда, по определению графа смежности, между ними не существует ребра. Если же одна из вершин не принадлежит исходному графу, то она совпадает с вершиной v($s = v$ или $r = v$), а значит, по определению множества $E' = E \cup \{\{u, w\} : w \in H\}$ и по построению множества H, в графе G' между ними не существует ребра.

Таким образом, любая пара вершин, пересечение нагрузок которых непустое, соединены магистралью, а вершины, нагрузки которых не пересекаются, не соединены ребром. Граф, обладающий таким свойством, является графом смежности, но не обязательно минимальным.

4.2.2 Алгоритм синтезирует минимальный граф смежности

Теорема 4. Алгоритм (листинг 3) синтезирует минимальный граф смежности.

Доказательство. Рассмотрим вторую часть алгоритма подробнее.

Сначала в множество вершин G' добавляются новая вершина v и все вершины исходного графа G (строка 3). Далее формируется минимальное множество вершин, добавление ребра к которым необходимо для сохранения свойств смежности, и для каждой такой вершины u (строка 4) в множество ребер E' добавляется соответствующее ребро $\{v, u\}$ (строка 5).

После добавления новой вершины и новых ребер формируется граф смежности G'. Далее, из графа G' необходимо удалить все те рёбра, отсутствие которых не приведёт к нарушению магистральных связей (строки 7–18). Данное преобразование описано, а также улучшено в [3, 4, 6, 21]. В его результате граф G' станет минимальным графом смежности.

5 Декрементальный алгоритм удаления вершины

Задан минимальный граф смежности G и одна из его вершин v; граф G и вершина v подаются на вход алгоритму. Требуется построить новый минимальный граф смежности — такой, что множество вершин состоит из разности множества вершин V графа G и выбранной вершины. Используя формальные обозначения, то же самое запишем следующим образом.

$G = \langle V, E \rangle$ — граф, подаваемый на вход алгоритму. $v \in V$ — вершина, которую нужно удалить. $G' = \langle V \backslash \{v\}, E' \rangle$, где E' — множество рёбер минимального графа смежности G'. Декрементальный алгоритм берёт за основу построения минимального графа смежности G' множество рёбер E — благодаря такому подходу предполагается, что декрементальный алгоритм будет работать быстрее жадного (прямого) алгоритма, так как большая или даже большая часть связей (рёбер) останется без изменений.

Напомним, что в контексте настоящей работы обозначения вершины и её нагрузки совпадают. Такой подход будет удобен для чтения и понимания графиков, представленных ниже, а также в разделе о корректности алгоритма.

5.1 Описание алгоритма

На листинге 4 приведён псевдокод декрементального алгоритма удаления вершины из минимального графа смежности.

Алгоритм состоит из двух частей. В первой части (строки 2–19) формируется граф смежности, с множеством вершин $V \setminus v$ и множеством рёбер, состяещем из множества рёбер исходного графа без рёбер, инцидентных вершине v, а также из всех возможных рёбер между вершинами таких рёбер (без v). Во второй части из вышеописанной вторичной структуры

Algorithm 4 Декрементальный алгоритм удаления вершины из минимального графа смежности.

input: $G = \langle V, E \rangle$, $v \in V$

output: $G' = \langle V', E' \rangle$

1: **function** DELETEINCREMENTAL
2: $E' = E$
3: $V' = V$
4: $E'' = \emptyset$
5: $V'' = \emptyset$
6: **foreach** (e **in** E)
7: **if** ($e.First == v$) **then**
8: $V'' = V'' \cup \{e.Second\}$
9: $E'' = E'' \cup \{e\}$
10: **if** ($e.Second == v$) **then**
11: $V'' = V'' \cup \{e.First\}$
12: $E'' = E'' \cup \{e\}$
13: **foreach** (x **in** V'')
14: **foreach** (y **in** V'')
15: **if** ($(W_x \cap W_y) \neq \emptyset$) **then**
16: $E' = E' \cup \{x, y\}$
17: $E' = E' \setminus E''$
18: $V' = V' \setminus \{v\}$
19: $G' = \langle V', E' \rangle$

20: **while** (**TRUE**)
21: $edge = \emptyset$
22: **foreach**(e **in** E')
23: **if** ($e.RemoveAllowed$) **then**
24: $edge = e$
25: **break foreach** //Выход из foreach
26: **if** ($edge == \emptyset$) **then**
27: **break while** //Выход из while
28: **if** (**PathExists**(G', $edge$, $edge.First$, $edge.Second$)) **then**
29: $E' = E' \setminus \{edge\}$
30: **else**
31: $edge.RemoveAllowed =$ **FALSE**

32: **return** $G' = \langle V', E' \rangle$

алгебраической байесовской сети удаляются рёбра, отсутствие которых не приведёт к нарушению магистральных связей (строки 20–31). Рассмотрим алгоритм более подробно.

В строках 4–12 формируется множество E'', состоящее из рёбер, инцидентных вершине v (строки 9, 12), а также множество вершин V'', состоящее из вершин рёбер E'', но не включающее в себя вершину v (строки 8, 11). Далее в множество E' добавляются всевозможные рёбра между вершинами из множества V'' (строка 16), пересечение нагрузок которых не пусто (строка 15). После этого формируется граф $G' = \langle V', E' \rangle$ (строка 19). Из данного графа необходимо удалить те ребра, отсутствие которых не приведёт к нарушению магистральных связей (строки 19–30). В результате такого преобразования граф G' станет минимальным графом смежности. Отметим, что рёбра удаляются с помощью жадного алгоритма, описанного в статье.

5.2 Корректность алгоритма

5.2.1 Алгоритм синтезирует граф смежности

Теорема 5. Вышеописанный алгоритм синтезирует граф смежности.

Доказательство. При добавлении допустимого ребра в минимальный граф смежности получается новый граф смежности, а удаление вершины и инцидентных ей рёбер с последущим добавлением допустимой клики на её место — переход от одного графа смежности к другому, с мешьшим числом вершин. Следовательно, после выполнения первой части алгоритма, граф G' будет графом смежности, при этом допустимость этих ребер обслувливается проверкой, осуществляемой в строке 15.

5.2.2 Алгоритм синтезирует минимальный граф смежности

Теорема 6. Алгоритм (листинг 4) синтезирует минимальный граф смежности.

Доказательство. Удаление ребра при сохранении магистральности есть не что иное, как переход от одного графа смежности к другому с уменьшением числа рёбер. В статьях было доказано, что множество графов смежности формирует матроид.

По свойствам матроида, последовательность допустимых удалений рёбер в графе смежности сходится, а полученный в результате завершившейся последовательности допустимых удалений граф будет минимальным графом смежности. Действительно, как было показано ранее, после выполнения первой части алгоритма, граф G' является графом смежности, а во второй части алгоритма происходит допустимое удаление рёбер. Таким образом, по свойствам матроида, алгоритм из листинга 4 на выходе формирует минимальный граф смежности, не включающий вершину v, что и требовалось доказать.

6 Система анализа и синтеза

Представленная в данной выпускной квалификационной работе система позволяет пользователям проводить анализ и синтез различых структур АБС, наглядно изучать поведение структур АБС при поступлении в сеть новых данных, а также проводить различного рода вычисления, важные в контексте вероятностных графических моделей. Разработанная платформа могла бы быть полезна для сотрудников математических кафедр университетов, занимающихся изучением АБС, как в качестве наглядного пособия, так и в качестве базовой библиотеки для дальнейших разработок. Также следует отметить, что данная система является результатом совместной разработки с Романовым Артемом Витальевичем и Березиным Алексеем

Ивановичем, однако в данной главе описан только мой вклад в данный комплекс программ.

В качестве языка программирования для реализации системы был выбран язык C# [26], поскольку на конечную систему налагался ряд требований, а именно: расширяемость и масштабируемость системы, высокая скорость работы технологии. Платформой для разработки была выбрана Microsoft Visual Studio 2015 [25]. Также использовался BitBucket [28], в качестве общего репозитория для совместной разработки системы.

6.1 Общие интерфейсы

Для совместной разработки системы были разработаны следующие интерфейсы: $IData$ и $ISeparable$, которые будут рассмотрены подробнее в следующих разделах. В качестве базового интерфеса графа был выбран интерфейс
$IBidirectionalGraph < TVertex, TEdge >$, реализованный в библиотеке с открытым исходным кодом QuickGraph [41]. Отмечу, что в контексте данной системы данный интерфейс, налагаемый на граф вторичной структуры выглядит так —
$IBidirectionalGraph < ISeparable < IData >, IEdge < ISeparable < IData >>>$, где $IEdge$ — интерфейс, также реализованный в библиотеке QuickGraph.

6.1.1 IData

Основной задачей данного интерфейса является обеспечение взаимодействия алгоритмов, отвечающих за генерацию и хранение вторичной структры и алгоритмов локального и глобального вывода. Также данный интерфейс позволяет абстрагироваться от конкретных типов данных, позволяя в дальнейшем работать с различными представлениями коньюнктов.

Листинг 1: Интерфейс IData.

```
public interface IData : IComparable, IComparable<IData>,
    IEquatable<IData>
    {
        string Data();
    }
```

6.1.2 ISeparable

Данный интерфейс представляет собой вершину графа.

Листинг 2: Интерфейс ISeparable<T>.

```
public interface ISeparable<T> : IComparable,
    IComparable<ISeparable<T>>, IEquatable<ISeparable<IData>>
    where T: IData
    {
        IEnumerable<T> ListOfData();
        int IntData();
        ISeparable<T> GetSeparator(ISeparable<T> separator);
        ISeparable<T> GetSeparator(IEnumerable<T> iteration);
        bool IsSubsetOf(ISeparable<T> other);
        bool IsSubsetOf(IEnumerable<T> other);
    }
```

6.2 Визуализация графа

Одной из подзадач данной работы является визуализация графа, поскольку все структуры АБС могут быть наглядно представлены в виде графа. Например: вторичная структура АБС сама по себе является графом, а вот первичная структура может быть представлена в виде максимального графа смежности, то есть графа смежности, с максимальным числом ребер.

В качестве основного инструмента визуализации графа, был выбран фреймворк с открытым исходным кодом Graph# [36], основанный на библиотеке QuickGraph [41]. Данный фреймворк включает в себя весь необходимый в контексте данной работы набор форм для отрисовки графа, однако не содержит никакой доступной докуметации, что существенно сказывается на времени освоения данной техологии.

Чтобы продемонстрировать всю простоту визуализации графа с помощью данного фреймворка, приведём пример кода:

Листинг 3: Визуализация графа.

```
1: <zoom:ZoomControl Mode="Original" Grid.RowSpan="2"
   Grid.ColumnSpan="4">
2:     <graph:GraphLayout
3:         Graph="{Binding GraphToVisuale ,Mode=OneWay}"
4:         LayoutAlgorithmType="{Binding SelectedItem}"
5:         OverlapRemovalAlgorithmType="FSA"
6:         HighlightAlgorithmType="Simple"
7:         >
8:     </graph:GraphLayout>
9: </zoom:ZoomControl>
```

На листинге 3 приведен отрывок XAML-кода, взятого из проекта, который используется для отображения графа. Рассмотрим подробнее: в строках 1–2 создаются два объекта — ZoomControl, который позволяет менять масштаб дочерних компонентов, и GraphLayout, отображающий граф. При этом, в строках 3–6 указываются параметры отображаемого графа, например: алгоритм расположения вершин на форме, макет графа и т.п. Конкретнее: в строке 3 граф, которой должен отображаться в форме graph (View часть паттерна MVVM), связывается (binding) с его представлением из ViewModel, также как и макет графа в строке 4.

6.3 Графический пользовательский интерфейс

Графический пользовательский интерфейс представляет собой набор диалоговых окон. Основой интерфейса является главное окно, представленное на рисунке 6.

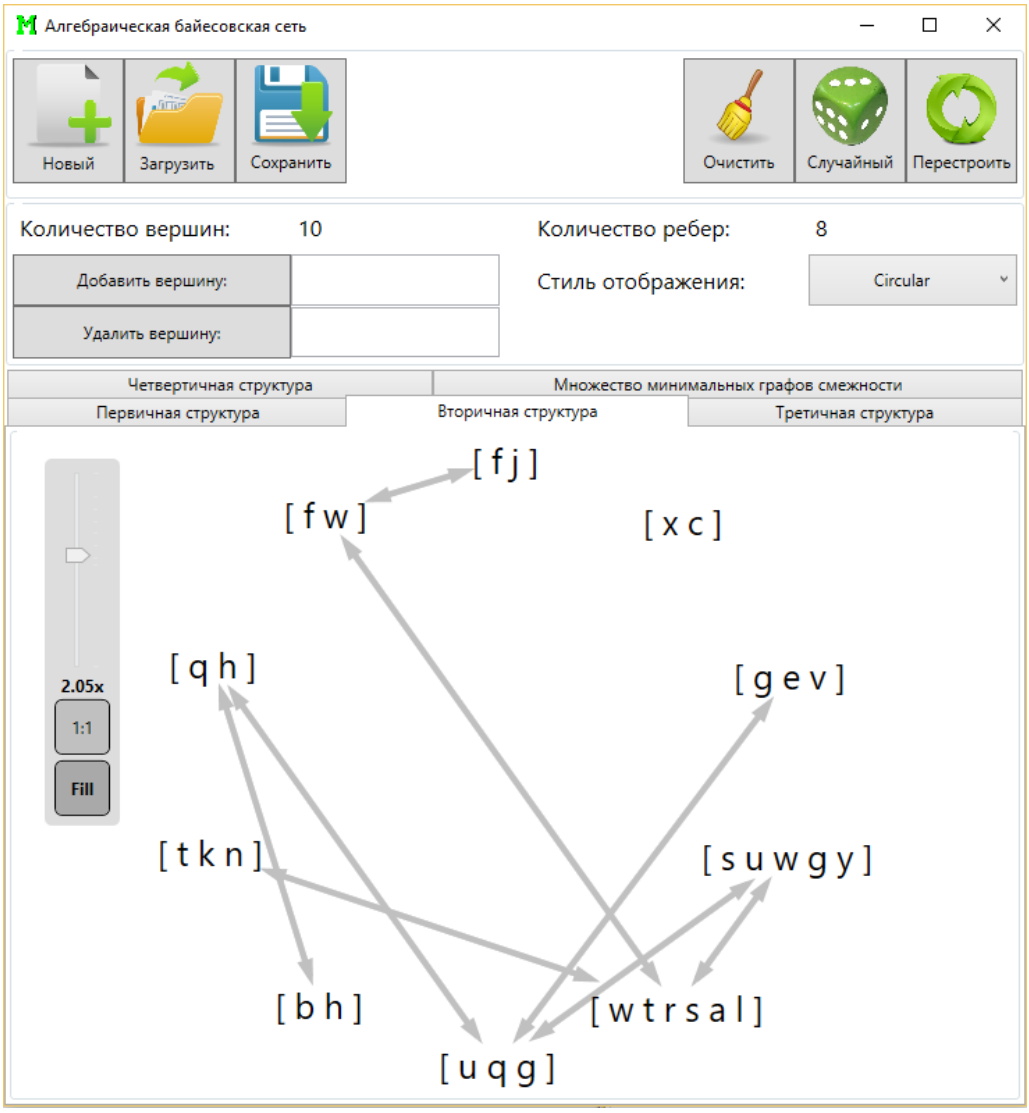


Рис. 6: Главное окно.

В главном окне располагаются несколько групп элементов, таких как «Панель для работы с файлами» или «Панель для работы с АБС», которые детально рассмотрены в следующих разделах.

6.3.1 Панель для работы с файлами

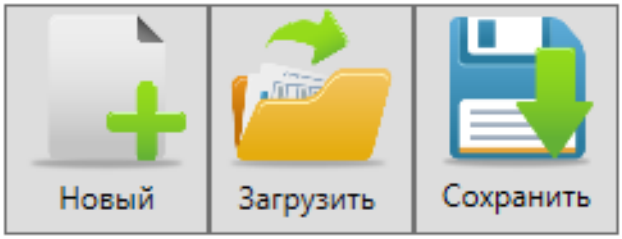


Рис. 7: Панель для работы с файлами.

Панель для работы с файлами представляет собой три кнопки со следующей функциональностью:

1) создание нового файла с пустой АБС;

2) загрузкой АБС из файла;

3) сохранением АБС в файл.

Иконки к кнопкам выбраны максимально похожие на соответствующие иконки из операционной системы Windows, чтобы пользователь интуитивно понимал их функциональность. Рассмотрим реализацию этих функциональностей подробнее, на примере кнопки сохранения:

Сохранение АБС в файл:

Поскольку MVVM-паттерн требует разделения модели и её представления — ниже приведен XAML-код кнопки сохранения АБС в файл, а также отрывок кода из ViewModel, реализующий её функциональность.

Листинг 4: SaveButton

```
<Button Grid.Column="2" Command="{Binding SaveCommand}"
    Height="90">
     <StackPanel HorizontalAlignment="Center"
         VerticalAlignment="Center">
          <Image Source="images\SaveImg.png" MinHeight="50"
              MinWidth="50" Height="64" Width="60"/>
```

```
        <TextBlock Text="{x:Static p:GraphResource.SaveString}"
            HorizontalAlignment="Center"
            VerticalAlignment="Top" MinHeight="20"/>
    </StackPanel>
</Button>
```

Листинг 5: SaveCommand

```
public void SaveCommandAction()
    {
        _stopSaving = false;
        var dlg = new Microsoft.Win32.SaveFileDialog
        {
            FileName = "Graph",
            DefaultExt = ".xml",
            Filter = "Xml documents (.xml)|*.xml"
        };
        var result = dlg.ShowDialog();
        if (result != true)
        {
            _stopSaving = true; return;
        }
        var filename = dlg.FileName;
        try
        {
            XmlTools.WriteToXml(filename, _abnData);
            _isDataDirty = false;
        }
        catch
        {
            MessageBox.Show("Error saving");
        }
    }
```

Листинг 6: Класс XmlTools

```
/// <summary>
```

```
/// Class XmlTools.
/// </summary>
public static class XmlTools {
    /// <summary>
    /// Convets to abn in XML.
    /// </summary>
    /// <param name="abn">The abn.</param>
    /// <returns>AbnInXml.</returns>
    private static AbnInXml ConvetToAbnInXml(AbnData abn) {
        var nodes =
            abn.SecondaryStructure.Vertices.Select(
                node => node.ListOfData.Select(data =>
                    data.GetData()).ToList())
                .Select(listOfData => new VertexInXml {Data =
                    listOfData})
                .ToList();
        return new AbnInXml {Vertices = nodes};
    }

    /// <summary>
    /// Convets to abn.
    /// </summary>
    /// <param name="overview">The overview.</param>
    /// <returns>AbnData.</returns>
    private static AbnData ConvetToAbn(AbnInXml overview) {
        var list =
            overview.Vertices.Select(vert =>
               vert.Data.Select(data => new
               StringData(data)).Cast<IData>().ToList())
                .Select(vertData => new
                    VertexData<IData>(vertData))
                .Cast<ISeparable<IData>>()
                .ToList();
        return new AbnData(list);
    }
```

```
    /// <summary>
    /// Reads from XML.
    /// </summary>
    /// <param name="path">The path.</param>
    /// <returns>AbnData.</returns>
    public static AbnData ReadFromXml(string path) {
        var reader =
            new System.Xml.Serialization.XmlSerializer(typeof
                (AbnInXml));
        using (var file = new System.IO.StreamReader(path)) {
            var overview = (AbnInXml) reader.Deserialize(file);
            return ConvetToAbn(overview);
        }
    }

    /// <summary>
    /// Writes to XML.
    /// </summary>
    /// <param name="path">The path.</param>
    /// <param name="abn">The abn.</param>
    public static void WriteToXml(string path, AbnData abn) {
        var writer = new
            System.Xml.Serialization.XmlSerializer(typeof
            (AbnInXml));
        using (var wfile = new System.IO.StreamWriter(path)) {
            writer.Serialize(wfile, ConvetToAbnInXml(abn));
        }
    }
}
```

6.3.2 Панель для работы с АБС

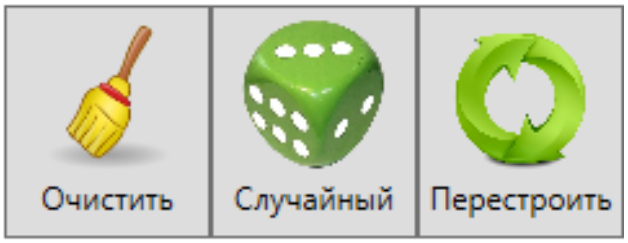


Рис. 8: Панель для работы с АБС.

Панель для работы с АБС также представлена тремя кнопками с функциональностью:

1) очищение АБС до пустой;

2) создание случайной АБС;

3) перестроение АБС.

Также как и в случае с кнопками из предыдущей панели, их функциональность будет рассмотрена подробнее:

Функциональность кнопки «Случайный»: иногда для проверки работоспособности или демонстрации системы требуется создать АБС и добавить в нее некоторое колличество вершин, однако в таких случаях удобнее машинно генерировать слуйчайную АБС, что и позвоялет делать эта кнопка.

Функциональность кнопки «Перестроить»: поскольку отображаемые графы не являются статическими, и пользователь может проводить с ними некоторые манипуляции (перемещение вершин, изменение масштаба и так далее), иногда может потребоваться получить их начальный вид и структуру, что и позволяет сделать данная кнопка.

6.3.3 Панель для работы с графом

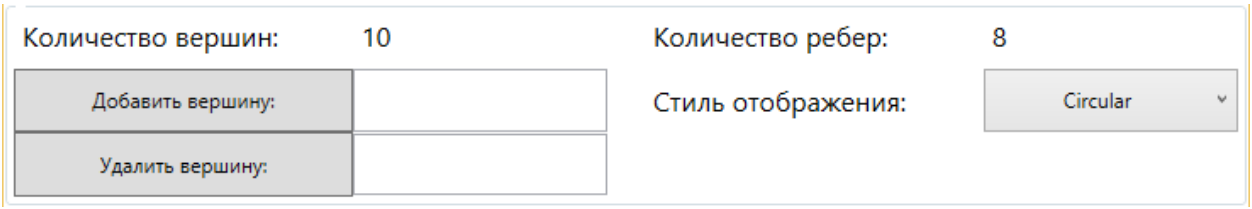


Рис. 9: Панель для работы с графом.

Панель для работы с графом содержит информацию о количестве вершин и ребер отображаемого графа, кнопки «Добавить вершину» и «Удалить вершину», а также соответствующие им поля ввода. Кроме этого панель содержит ComboBox, позволяющий пользователю выбирать стиль отображения графа, например: Circular или Tree.

6.3.4 Панель структур АБС

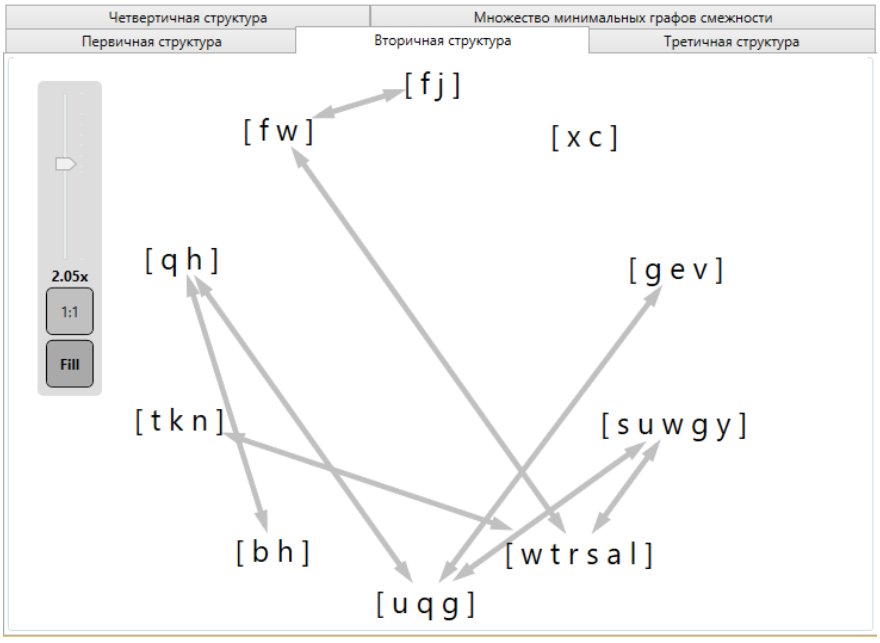


Рис. 10: Панель структур АБС.

Панель структур АБС: состоит из набора вкладок (tab items) для каждой глобальной структуры АБС, на которых отоброжается информация о соответствующей структуре АБС:

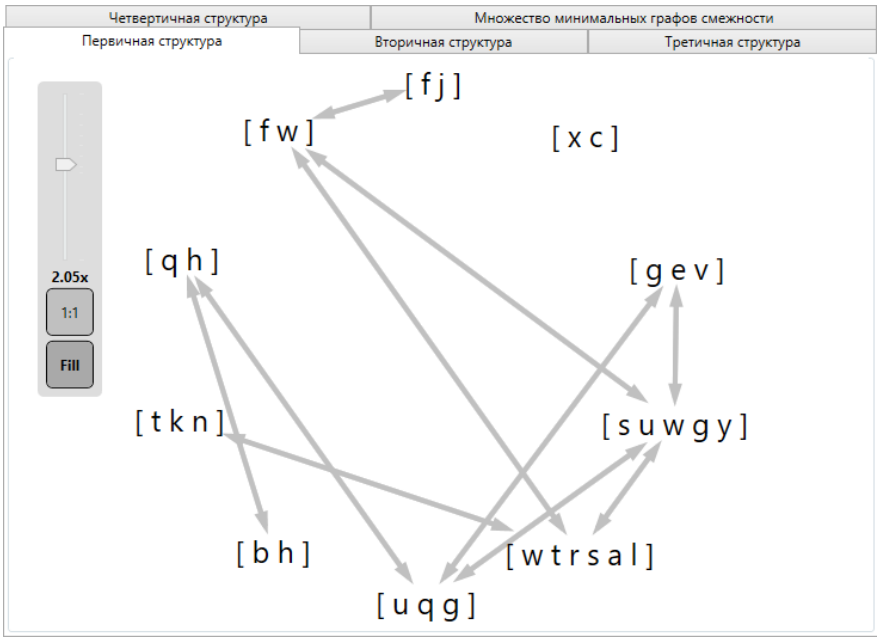


Рис. 11: Вкладка «Первичная структура».

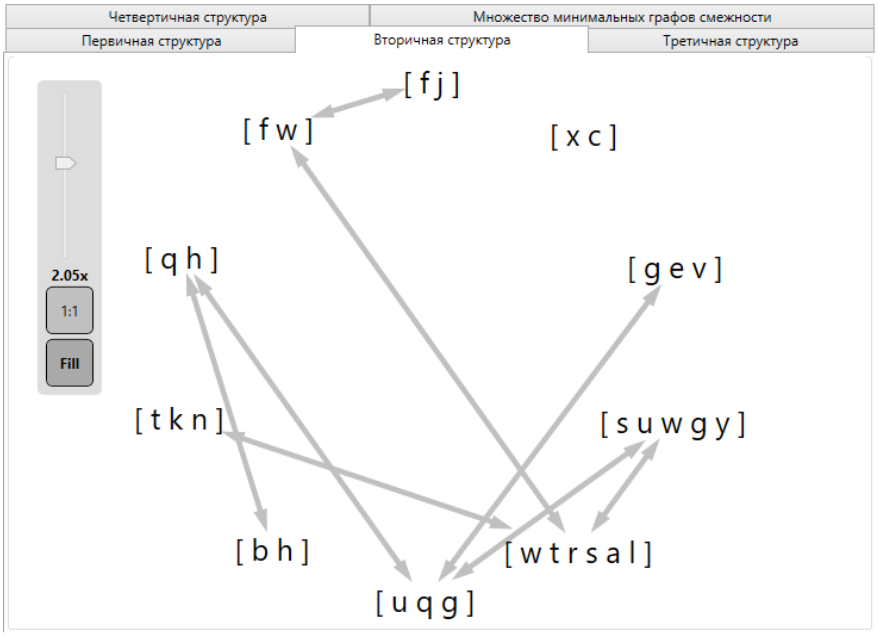


Рис. 12: Вкладка «Вторичная структура».

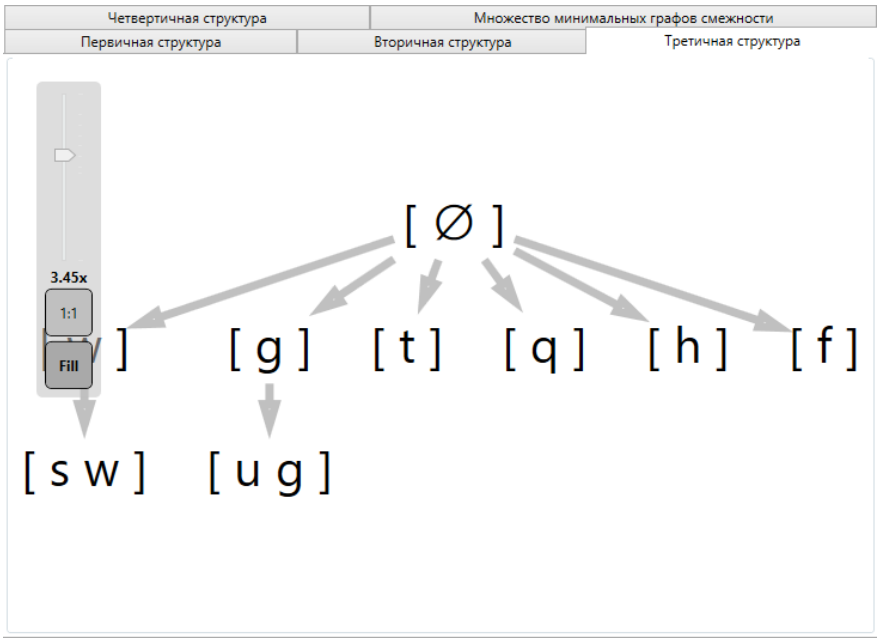


Рис. 13: Вкладка «Третичная структура».

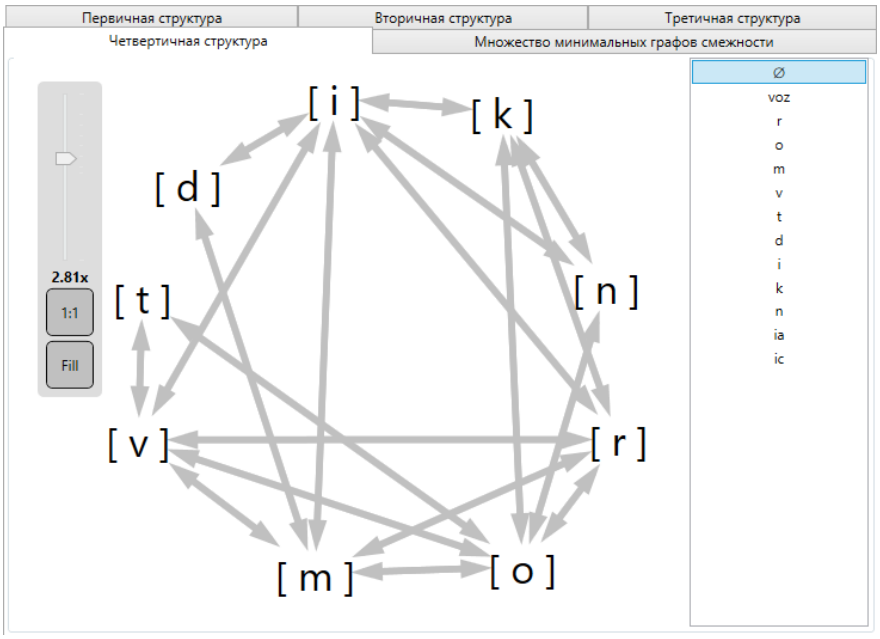


Рис. 14: Вкладка «Четвертичная структура».

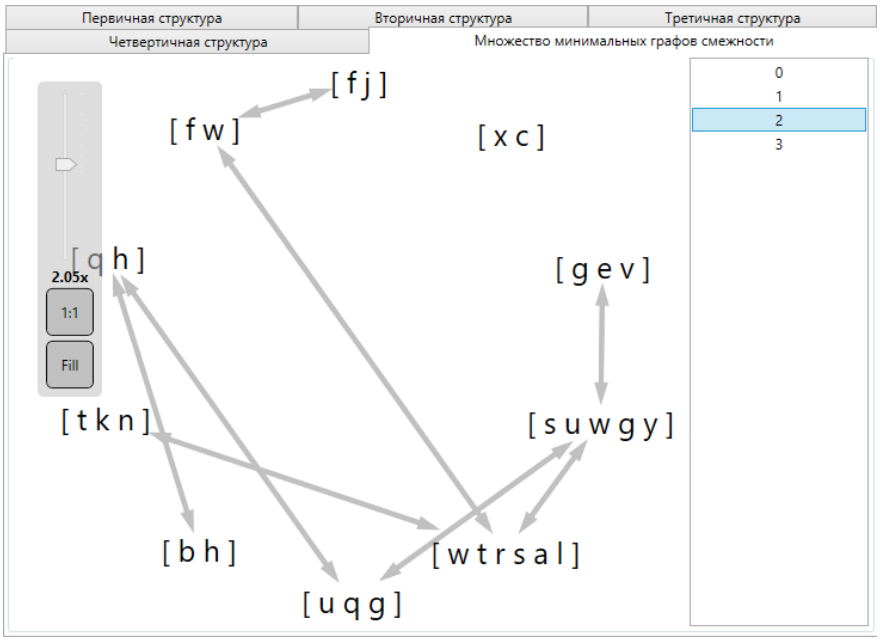


Рис. 15: Вкладка «Множество минимальных графов смежности».

1. Вкладка «Первичная структура» (рис. 11).
2. Вкладка «Вторичная структура» (рис. 12).
3. Вкладка «Третичная структура» (рис. 13).
4. Вкладка «Четвертичная структура» (рис. 14).
5. Вкладка «Множество минимальных графов смежности» (рис. 15).

Вкладка «Первичная структура» состоит из GraphLayout-а, отображающего первичную структуру АБС в виде максимального графа смежности, а также ZoomControl-а, позволяющего менять масштаб этого графа. Такой нестандартный способ отображения первичной структуры был выбран из-за большей наглядности по сравнению с традиционным представлением первичной структуры в виде списка вершин.

Вкладка «Вторичная структура» также содержит GraphLayout и ZoomControl, отвечающие за отображение вторичной структуры АБС — минимального графа смежности. Отметим, что множество минимальных

графов смежности [18, 23], зачастую, имеет мощность больше еденицы, и выбор одного из таких графов не является принципиальным. Однако алгоритмы локального и глобального вывода могут работать быстрее или медленнее на разных элементах такого множества, но явная зависимость от конкретных характеристик вторичной структуры пока не выявлена, и нуждается в дальнейшем изучении, не входящем в рамки текущей работы.

Вкладка «Третичная структура» отличается от предыдущей вкладки лишь тем, что GraphLayout отображает третичную структуру АБС, используя алгоритм отображения графа «Tree».

Вкладка «Четвертичная структура» состоит из списка графов, именнованых по их сепаратору, а также GraphLayout-а, отображающего выбранный граф.

Вкладка «Множество минимальных графов смежности» также состоит из списка графов и GraphLayout-а, отображающего выбранный граф.

Заключение

В данной работе рассмотрены три алгоритма, существенно ускоряющие синтез [2,5] нового минимального графа смежности при изменении первичной структуры исходного минимального графа смежности.

Вышеуказанные алгоритмы особенно удобны для интерактивных программных систем, когда минимальный граф смежности строится в диалоге с пользователем. Последний ожидает от системы достаточного быстродействия, полагая, что небольшое изменение в данных, в частности, при добавлении или удалении вершины, влечет умеренное изменение минимального графа смежности.

Разработана система анализа и синтеза вторичной структуры АБС, реализующая все предъевленные к ней требования.

Все задачи выполнены. Все цели достигнуты.

Список литературы

[1] Золотин А.А., Тулупьев А.Л., Сироткин А.В. Матрично-векторные алгоритмы нормировки для локального апостериорного вывода в алгебраических байесовских сетях // Научно-технический вестник информационных технологий, механики и оптики. 2015. Том 15. № 1. С. 78–85.

[2] Зотов М.А., Левенец Д.Г., Тулупьев А.Л., Золотин А.А. Синтез вторичной структуры алгебраических байесовских сетей: инкрементальный алгоритм и статистическая оценка его сложности // Научно-технический вестник информационных технологий, механики и оптики. 2016. № 1. С. 122–132.

[3] Зотов М.А., Тулупьев А. Л. Синтез вторичной структуры алгебраических байесовских сетей. // Компьютерные инструменты в образовании(2015. Выпуск 1).

[4] Зотов М.А., Тулупьев А.Л., Сироткин А.Л. Статистические оценки сложности прямого и жадного алгоритмов синтеза вторичной структуры алгебраических байесовских сетей // Нечеткие системы и мягкие вычисления. 2015. Т. 10. №1. С. 75–91

[5] Левенец Д.Г., Зотов М.А., Тулупьев А.Л. Инкрементальный алгоритм синтеза минимального графа смежности. // Компьютерные инструменты в образовании. 2015. № 6. С. 3–18.

[6] Опарин В.В., Тулупьев А.Л. Синтез графа смежности с минимальным числом ребер: формализация алгоритма и анализ его корректности. // Тр. СПИИРАН. 2009. №11. С. 142–157.

[7] Опарин В.В., Фильченков А.А., Сироткин А.В., Тулупьев А.Л. Матроидное представление семейства графов смежности над набором фрагментов знаний // Научно-технический вестник Санкт-Петербургского

государственного университета информационных технологий, механики и оптики. 2010. №4(68). С. 73–76.

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

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

[10] Тулупьев А.Л. Алгебраические байесовские сети: локальный логико-вероятностный вывод: // Учеб. пособие. СПб.: СПбГУ; ООО Издательство «Анатолия», 2007. С. 80. (Сер. Элементы мягких вычислений)

[11] Тулупьев А.Л. Алгебраические байесовские сети: открытые вопросы локального автоматического обучения // СПИСОК-2014: Материалы всероссийской научной конференции по проблемам информатики. Санкт-Петербург, 2014. С. 569–577.

[12] Тулупьев А.Л. Вероятностная логика и вероятностные графические модели в базах фрагментов знаний с неопределенностью // Интегрированные модели, мягкие вычисления, вероятностные системы и комплексы программ в искусственном интеллекте. Научно-практическая конференция студентов, аспирантов, молодых ученых и специалистов (Коломна, 26–27 мая 2009 г.). Научные доклады. В 2-х т. Т. 1. М.: Физматлит, 2009. С. 26–46.

[13] Тулупьев А.Л. Дерево смежности с идеалами конъюнктов как ациклическая алгебраическая байесовская сеть // Тр. СПИИРАН. Вып. 3, т. 1. СПб.: Наука, 2006. С. 198–227.

[14] Тулупьев А.Л., Николенко С.И., Сироткин А.В. Байесовские сети: логико-вероятностный подход. СПб.: Наука, 2006. С. 607.

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

[16] 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. С. 400.

[17] Фильченков А.А. Синтез графов смежности в машинном обучении глобальных структур алгебраических байесовских сетей. Дисс.... к-та физ.-мат. н. Самара, 2013. С. 339. (Самарск. гос. аэрокосм. ун-т им. ак. С.П. Королева (нац. исслед.))

[18] Фильченков А.А., Мусина В.Ф., Тулупьев А.Л. Алгоритм рандомизированного синтеза минимального графа смежности // Тр. СПИИРАН. 2013. Вып. 2(35). С. 221–234. ун-т.

[19] Фильченков А.А., Тулупьев А.Л. Анализ циклов в минимальных графах смежности алгебраических байесовских сетей // Тр. СПИИРАН. 2011. №17. С. 151–173.

[20] Фильченков А.А., Тулупьев А.Л. Структурный анализ систем минимальных графов смежности // Тр. СПИИРАН. 2009. №11. С. 104–129.

[21] Фильченков А.А., Тулупьев А.Л., Сироткин А.В. Минимальные графы смежности алгебраической байесовской сети: нормализация основ

синтеза и автоматического обучения // Нечеткие системы и мягкие вычисления. 2011. Т. 6. № 2. С. 145-163.

[22] Фильченков А.А., Тулупьев А.Л., Сироткин А.В. Особенности анализа вторичной структуры алгебраической байесовской сети // Труды СПИИРАН. 2010. № 1 (12). С. 97-118. (цит. 24, ИФ РИНЦ)

[23] Фильченков А.А., Фроленков К.В., Сироткин А.В., Тулупьев А.Л. Система синтеза подмножеств минимальных графов смежности // Тр. СПИИРАН. 2013. Вып. 4(27). С. 200–244.

[24] Шинкаренко В.И. Зависимость временной эффективности алгоритмов и программ обработки больших объемов данных от их кэширования // Математические машины и системы. 2007. №2. С. 43–55.

[25] Обзор продуктов Visual Studio 2015 [Электронный ресурс]. Режим доступа: URL: https://www.visualstudio.com/ru-ru/products/vs-2015-product-editions.aspx (дата обращения 07.05.2016).

[26] Руководство по программированию на C# [Электронный ресурс]. Режим доступа: URL: https://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx (дата обращения 07.05.2016).

[27] Сайт компании Microsoft [Электронный ресурс]. Режим доступа: URL: https://www.microsoft.com/ru-ru/ (дата обращения 07.05.2016).

[28] Сайт проекта BitBucket [Электронный ресурс]. Режим доступа: URL: https://bitbucket.org/ (дата обращения 07.05.2016).

[29] Barrett C., Marathe A., Marathe M., Cook D., Hicks G., Faber V., Srinivasan A., Sussmann Y., Thornquist H. Statistical Analysis of Algorithms: A Case Study of Market-Clearing Mechanisms in the Power Industry. Journal of Graph Algorithms and Applications (JGAA). 2003. Vol. 7. P. 3–31.

[30] Jaynes E.T. Bayesian Methods: General Background // Maximum-Entropy and Bayesian Methods in Applied Statistics, by J. H. Justice (ed.). Cambridge: Cambridge Univ. Press, 1986. P. 1–19.

[31] Kas M., Wachs M., Carley K.M., Carley.L.R. Incremental Algorithm for Updating Betweenness Centrality in Dynamically Growing Networks. // Ozyer, T and Carrington, P, editor,2013 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM), p.39–46, 345 E 47TH ST, Newyork, NY 10017 USA, 2013. IEEE. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Niagara Falls, Canada, aug. 25–28, 2013.

[32] Maier D. Theory of Relational Databases. // Rockville, MD: Computer Science Press, 1983. P. 637

[33] Pearl J. Causality: Models, Reasoning, and Inference. Cambridge: Cambridge University Press, 2000. P. 400.

[34] Respondek J. S. Dynamic data structures in the incremental algorithms operating on a certain class of special matrices //Computational Science and Its Applications–ICCSA 2014. – Springer International Publishing, 2014. – C. 171-185.

[35] Song X., Yu L., Sun H. An Incremental Query Algorithm for Optimal Path Queries Under Traffic Jams. // Yu, F and Chen, W and Chen, Z and Yuan, J, editor, ISCSCT 2008: International Symposiumon Computer Science and Computational Technology, Proceedings, p.472–475, 10662 Los Vaqueros Circle, PO BOX3014, Los Alamitos, CA 90720-1264 USA, 2008. IEEE Computer SOC. International Symposium on Computer Science and Computational Technology, Shanghai, Peoples R China, dec.20–22, 2008.

[36] Graph# [Электронный ресурс]. Режим доступа: URL: https://graphsharp.codeplex.com/ (дата обращения 07.05.2016).

[37] Incremental algoritms [Электронный ресурс]. Режим доступа: URL: http://c2.com/cgi/wiki? IncrementalAlgorithms (дата обращения 01.11.2015).

[38] Microsoft Silverlight [Электронный ресурс]. Режим доступа: URL: https://www.microsoft.com/silverlight/ (дата обращения 07.05.2016).

[39] Introduction to Model/View/ViewModel pattern for building WPF apps [Электронный ресурс]. Режим доступа: URL: https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/ (дата обращения 07.05.2016).

[40] MSDN: Windows Presentation Foundation [Электронный ресурс]. Режим доступа: URL: https://msdn.microsoft.com/ru-ru/library/ms754130.aspx (дата обращения 07.05.2016).

[41] QuickGraph, Graph Data Structures And Algorithms for .NET [Электронный ресурс]. Режим доступа: URL: https://quickgraph.codeplex.com/ (дата обращения 07.05.2016).

[42] Wikipedia: Model-View-ViewModel [Электронный ресурс]. Режим доступа: URL: https://ru.wikipedia.org/wiki/Model-View-ViewModel (дата обращения 07.05.2016).

[43] Wikipedia: Windows Presentation Foundation [Электронный ресурс]. Режим доступа: URL: https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation (дата обращения 07.05.2016).

[44] Wikipedia: XAML [Электронный ресурс]. Режим доступа: URL: https://ru.wikipedia.org/wiki/XAML (дата обращения 07.05.2016).

[45] .NET Framework [Электронный ресурс]. Режим доступа: URL: https://www.microsoft.com/net/default.aspx (дата обращения 07.05.2016).