Санкт-Петербургский государственный университет

Фундаментальная информатика и информационные технологии
Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Иванова Анна Валерьевна

Алгебраические байесовские сети: визуализация глобальных структур и фронтэнд системы их обработки (проектная работа)

Выпускная квалификационная работа

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

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

Санкт-Петербург
2017

SAINT PETERSBURG UNIVERSITY

Fundamental Computer Science and Information Technologies
Mathematical and Software Support for Computers, Computer Systems and Networks

Anna Ivanova

Algebraic Bayesian Networks: visualisation of global structures and front-end systems processing thereof (joint project)

Graduation Thesis

Scientific supervisor:
Professor, D. Sc., PhD, Assoc. Prof. Alexander Tulupyev

Reviewer:
Assoc. Prof. NRU HSE, PhD Alexander Sirotkin

Saint Petersburg
2017

Оглавление

Введение **4**

1. Обзор **7**

1.1. Введение 7
1.2. Актуальность работы 7
1.3. Цели и задачи 9
1.4. Выводы к главе 10

2. Алгебраические байесовские сети: структура и основные определения **11**

2.1. Введение 11
2.2. АБС: основные определения 11
2.3. Алгоритмы синтеза минимального графа смежности . . 14
2.4. Программные технологии 22
2.5. Выводы к главе 26

3. Алгоритмы синтеза МГС: сравнительный анализ **27**

3.1. Введение 27
3.2. Описание вычислительных экспериментов 27
3.3. Результаты вычислительных экспериментов 30
3.4. Проверка данных на нормальность 35
3.5. Выводы к главе 37

4. Визуализация структур АБС **38**

4.1. Введение 38
4.2. Визуализация компонентов АБС 39
4.3. Примеры работы 43
4.4. Выводы к главе 46

Заключение **47**

Список литературы **50**

Введение

Значительным достижением в развитии информатики и информационных систем стало появление так называемых экспертных систем. Экспертная система представляет собой программу, имеющую на входе некоторый набор знаний предметной области и способную на основе предоставленной информации сделать вывод о состоянии некоторых объектов системы и системы в целом. Зачастую, подобные системы обладают значительной долей неопределенности. Такая неопределенность может быть вызвана как недостатком знаний, так и сложностью перевода высказываний в язык, понятный ЭВМ.

Вероятностные графические модели (ВГМ) являются одной из моделей описания экспертной системы, способной работать в условиях неопределенности. Если предположить, что внутри предметной области возможна декомпозиция на небольшие утверждения, определения, объекты, называемые фрагментом знаний, то можно сказать, что все ВГМ представляют собой базы фрагментов знаний с неопределенностью [26]. К различным видам ВГМ можно отнести марковские сети, байесовские сети доверия, а также алгебраические байесовские сети [29, 33, 37].

Алгебраические байесовские сети (АБС) являются одним из представителей класса вероятностных графических моделей [4]. АБС способны обрабатывать большие наборы данных с неопределенностью, декомпозируя данные на небольшие фрагменты, состоящие из тесно связанных между собой частей.

В качестве математической модели фрагмента знаний в теории АБС используется идеал конъюнктов с оценками вероятности истинности элементов. Оценки могут быть как скалярными, так и интервальными. Наличие возможности использования интервальных оценок особенно важно при обработке данных с неопределенностью и несогласованных данных. В качестве вторичной структуры АБС выступают графы смежности.

На данный момент теория АБС была значительно развилась. Был проведен ряд работ, связанных со структурами АБС, а также работ,

которые развивают различные подходы логико-вероятностного вывода в АБС [9, 21, 23]. Работы в данной области продолжаются и в нынешнее время. Отдельным классом алгоритмов являются алгоритмы, генерирующие множества вторичной структуры или множества минимальных графов смежности. Результаты работ над синтезом вторичной структуры можно изучить в [9, 30]. Основная информация о таких структурах, как третичная и четвертичная доступна в [38, 40].

В связи с развитием теории возникла острая необходимость в создании программного комплекса, реализующего математические результаты и позволяющего визуализировать структуры АБС, а также проводить вычислительные эксперименты над ними.

Ранее для работы с алгебраическими байесовскими сетями были разработаны следующие библиотеки: в 2009 году была разработана java-библиотека AlgBN Modeler j.v.01 [36], а в 2011 году была разработана библиотека AlgBN KPB Reconciler cpp.v.01 [35] на языке C++. Данные библиотеки поддерживают функциональность, необходимую для работы с АБС. Однако, развитие теории в последние годы показывает необходимость в новом программном комплексе, создание которого нацелено на использование последних алгоритмов и предполагает направленность на увеличение точности результата, а также улучшение времени обработки данных [1, 2, 3, 7].

Данная работа является частью более широких инициативных проектов, выполняющихся в лаборатории теоретических и междисциплинарных основ информатики СПИИРАН под руководством А.Л. Тулупьева; кроме того, разработки были частично поддержаны грантами РФФИ 15-01-09001-а — «Комбинированный логико-вероятностный графический подход к представлению и обработке систем знаний с неопределенностью: алгебраические байесовские сети и родственные модели», 12-01-00945-а — «Развитие теории алгебраических байесовских сетей и родственных им логико-вероятностных графических моделей систем знаний с неопределенностью»,09-01-00861-а — «Методология построения интеллектуальных систем поддержки принятия решений на основе баз фрагментов знаний с вероятностной неопределенностью», а также

проектом по ФЦНТП «Исследования и разработки по приоритетным направлениям развития науки и техники на 2002–2006 годы», 2006-РИ-19.0/001/211,Государственный контракт от «28» февраля 2006 г., № 02.442.11.7289, «Направленные циклы в байесовских сетях доверия: вероятностная семантика и алгоритмы логико-вероятностного вывода для программных комплексов с байесовской интеллектуальной компонентой».

1. Обзор

1.1. Введение

Данная глава представляет собой обзор существующих комплексов программ, предоставляющих функциональность для работы с алгебраическими байесовскими сетями. Обосновывается актуальность выбранной темы, выявляется цель работы и ставятся задачи для ее достижения.

1.2. Актуальность работы

В настоящий момент существует ряд программных решений, позволяющих осуществлять работу с алгебраическими байесовскими сетями. К таким решениям относятся в частности две библиотеки, написанные на Java и на C++ [36, 35], которые включают в себя формирование ФЗ различных видов, их обработку, средства для локального априорного и апостериорного вывода и средства для глобального вывода. Первая библиотека AlgBN Modeler j.v.01 была разработана на Java в 2009 году [36]. Основная особенность данной реализации – разделение такого функционала, как хранение фрагментов знаний и инструментов локального и глобального вывода. Данный подход позволяет формировать и хранить ФЗ независимо от применяемых инструментов вывода. Данная реализация содержит в себе ключевые алгоритмы логико – вероятностного вывода, а именно: локальная проверка и поддержание непротиворечивости, а также априорный и апостериорный вывод. Вторая библиотека AlgBN KPB Reconciler cpp.v.01 была разработана на C++ [35]. В реализации данной библиотеки применяется иной подход, который подразумевает объединение средств формирования и обработки фрагментов знаний с инструментами локального вывода.

Следует отметить, что ранее для работы с алгебраическими байесовскими сетями в лаборатории СПИИРАН было также разработано десктопное приложение под руководством профессора А.Л. Тулупьева [17, 18]. Приложение обладает значительным функционалом, однако

содержит ряд недостатков.

Данные программные комплексы были направлены на реализацию структурных и алгоритмических аспектов, однако оказались недостаточно адаптированы для применения в учебном процессе, так как их использование подразумевает частый перенос и неоднократную установку большого объема программного обеспечения. Также, работа программного комплекса на платформах, отличных от Windows не была гарантирована.

Таким образом, актуальной задачей является создание веб - приложения, предоставляющего возможности работы с алгебраическими байесовскими сетями. Преимуществом такого решения является наличие возможности работы с актуальным вариантом программного комплекса обработки АБС в любое время в условиях стабильного интернет-соединения. К возможностям работы с АБС относятся синтез структур АБС, а также осуществление машинного обучения на данной базе фрагментов знаний. В рамках реализации такого приложения, возникает необходимость в визуализации глобальных структур алгебраической байесовской сетей, их создании и параметризации.

Стоит отметить, что реализация программного комплекса обработки АБС в виде веб-приложения позволит устранить следующие недостатки десктопной реализации: отсутствие кроссплатформенности и необходимости полного переноса комплекса программ.

В процессе работы с АБС пользователь рассматривает как численные показатели составляющих сети, так и ее графическое представление. Правильных подход к визуализации АБС может повлиять на скорость принятия решения пользователем.

Основными объектами АБС, с которыми работает пользователь, являются первичная и вторичная структуры. Соответственно важной частью процесса визуализации работы с АБС является создание визуальных представлений данных структур.

Одной из математических моделей вторичной структуры АБС является минимальный граф смежности. Его использование обусловлено тем, что для ряда алгоритмов необходимым требованием является

представление вторичной структуры в виде ациклического минимального графа смежности [28, 30]. Ранее были предложены прямой и жадный алгоритмы синтеза минимального графа смежности [20, 38]. В силу того, что в процессе применения данных алгоритмов синтезируются новые структуры, актуальной задачей является создание инкрементального алгоритма, который преобразовывал бы уже существующую структуру.

Чтобы обеспечить ускорение работы программного комплекса по сравнению с другими реализациями, необходимо провести сравнительный анализ существующих алгоритмов синтеза минимального графа смежности. Для выявления наиболее предпочтительных для использования в отдельных случаях алгоритмов требуется провести ряд статистических экспериментов и доказать их корректность.

1.3. Цели и задачи

Объектом исследования являются *алгебраические байесовские сети*, а предметом исследования — *алгоритмы синтеза вторичной структуры*.

Целью проекта является *автоматизация обработки алгебраических байесовских сетей*. Целью данной работы является *визуализация глобальных структур и автоматизация анализа алгоритмов их синтеза*.

Цели достигаются путем решения следующих задач:

1) проведение вычислительных экспериментов над конкурирующими алгоритмами синтеза МГС;

2) обоснование корректности вычислительных экспериментов;

3) разработка алгоритмы визуализации первичной и вторичной структур АБС;

4) обеспечение интеграцию с другими частями системы;

5) создание ряда примеров, демонстрирующих работу системы.

1.4. Выводы к главе

В данной главе была обоснована актуальность работы, сформулирована цель работы, а также выявлены задачи, необходимые для ее достижения.

2. Алгебраические байесовские сети: структура и основные определения

2.1. Введение

Важным отличием алгебраических байесовских сетей от родственного аппарата — *Байесовских сетей доверия* является возможность работы с интервальными оценками [26, 33]. Также, АБС обладает аппаратом, позволяющим осуществлять поддержку непротиворечивости сети, априорный и апостериорный логико-вероятностный вывод [22, 32, 27, 31].

Основными структурами в теории АБС являются первичная и вторичная структуры. Множество фрагментов знаний некоторой предметной области является первичной структурой, а вторичная структура определяет связь между фрагментами знаний.

Для осуществления процесса визуализации в работе над программным комплексом необходимо иметь представление об основных структурах АБС. В данной главе вводятся основные понятия и определения, описывающие фрагмент знаний в АБС, вводятся понятия первичной и вторичной структур, а также рассматриваются существующие алгоритмы синтеза минимального графа смежности, являющегося математической моделью вторичной структуры.

2.2. АБС: основные определения

2.2.1. Первичная структура

Для представления знаний предметной области и связей между ними в теории алгебраических байесовских сетей используется декомпозиция исходных данных на фрагменты знаний [34] . Под фрагментом знаний подразумевается множество близко связанных между собой утверждений. При этом сами фрагменты знаний могут быть слабо связанными. Поскольку эксперты в предметной области обычно задают зависимости между несколькими атомарными утверждениями, разбиение на

фрагменты знаний используется для уменьшения количества вычисляемых вероятностей.

Для формального описания алгебраической байесовской сети, являющейся моделью знаний с неопределенностью, введем математический аппарат. Следующие определения и понятия основываются на ряде источников [25, 26, 33, 42].

Алфавит атомов представляет собой набор элементарных высказываний экспертов о предметной области.

Определение 2.1 *[25] Алфавит атомов* $A = x_i{}_{i=0}^{n-1}$ *— конечное множество атомарных пропозициональных формул.*

Определение 2.2 *[25] Конъюнкцию некоторого числа атомарных переменных вида:* $x_{i_1} \wedge x_{i_2} \wedge ... \wedge x_{i_k}$*, назовем конъюнктом или цепочкой конъюнкций.*

Определение 2.3 *[26] Идеалом конъюнктов являются цепочки следующего вида:* $\{x_{i_1}x_{i_2}...x_{i_k} | 0 \leq i_1 < i_2 < ... < i_k \leq n-1, 0 \leq k \leq n\}$.

В данном определении $x_{i_1}x_{i_2}...x_{i_k}$ является цепочкой конъюнкций с опущеным для удобства знаком конъюнкции.

Определение 2.4 *[25] Упорядоченную пару* $\langle \zeta, p \rangle$*, где* ζ *— идеал конъюнктов над множеством атомов* $S = \{x_{i_1}x_{i_2}...x_{i_k}\}$*, для каждого элемента которого определена функция* p *из* ζ *в интервал* $[0; 1]$ *назовем математической моделью фрагмента знаний с неопределенностью* ζ*.*

Носителем называется идеал конъюнктов, над которым строится фрагмент знаний.

Алгебраическая байесовская сеть характеризуется двумя объектами: первичной и вторичной структурами [25, 41]. Множество построенных над подмножествами атомов заданного алфавита и связанные между собой фрагменты знаний называется первичной структурой. Связь между фрагментами знаний существует, если пересечение алфавитов, над которыми они построены, не пусто. Пересечение фрагментов знаний также является фрагментом знаний и построено оно над пересечением алфавитов исходных фрагментов знаний.

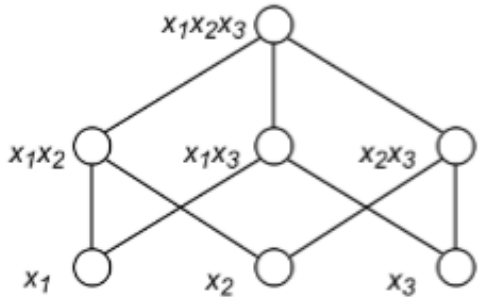

Рис. 1: [26] Фрагмент знаний, построенный над алфавитом $\{x_1, x_2, x_3\}$

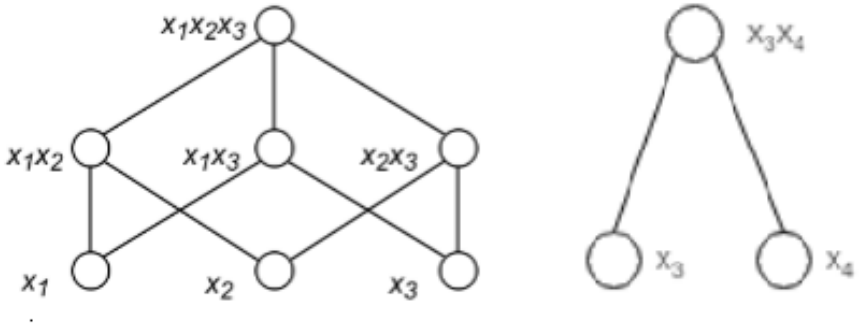

Рис. 2: [26] Пример первичной структуры АБС, состоящей из двух ФЗ, построенных над $\{x_1, x_2, x_3, x_4\}$

2.2.2. Вторичная структура

Вторичная структура АБС описывает связь между фрагментами знаний. При этом формируется граф, вершинами которого являются фрагменты знаний, а ребрами выступают связи между ними.

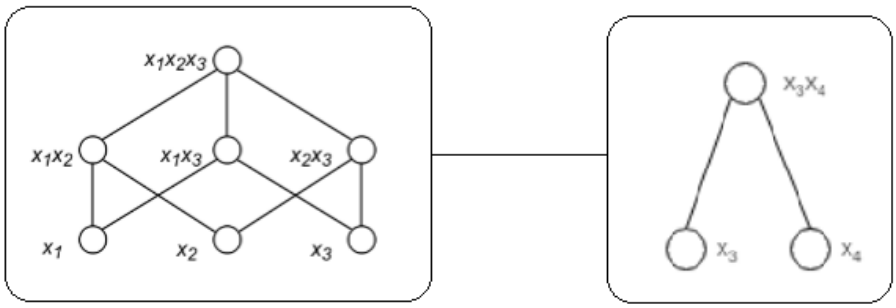

Рис. 3: Пример вторичной структуры АБС, состоящей из двух ФЗ, построенных над $\{x_1, x_2, x_3, x_4\}$

Вторичной структурой АБС является граф смежности (или дерево смежности [30] в условиях отсутствия циклов в графе) с фрагментами

знаний в узлах. Множество вторичных структур можно построить по каждой первичной структуре. Подробно подходы к построению таких структур рассмотрены в [34, 39, 42].

Определение 2.5 *[42] Граф смежности это ненаправленный связный граф, для которого выполняются следующие условия:*

1) *между каждой парой узлов, веса которых содержат общие элементы, существует путь без самопересечений, такой, что в веса каждого из узлов этого пути входят все элементы, общие для начального и конечного узлов;*

2) *вес одного узла не входит полностью в вес никакого другого узла.*

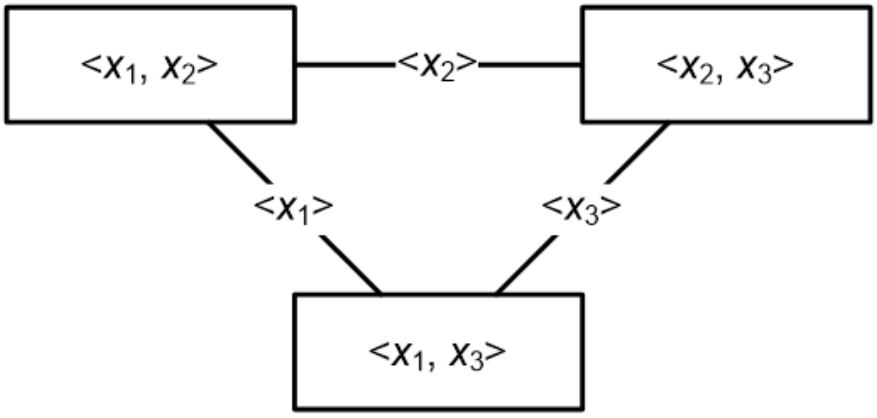

Рис. 4: Граф смежности

Более формальное определение будет дано в следующем разделе.

2.3. Алгоритмы синтеза минимального графа смежности

Для реализации процесса синтеза минимального графа смежности был предложен ряд алгоритмов. Данные алгоритмы отличается друг от друга как возможностями применения в различных ситуациях, так и скоростью выполнения.

Ранее были предложены прямой и жадный алгоритмы синтеза [20, 38]. Однако они базируются на некоторых скрытых свойствах минимального графа смежности, а также свойствах, которые становятся известными только после того, как искомый граф построен. Поэтому в качестве конкурирующего алгоритма был предложен инкрементальный алгоритм [8].

2.3.1. Основные определения

Сформиулируем основные определения, сложившиеся в [19, 24, 38], которые будут использоваться далее.

Определение 2.6 *[19]Граф* $G = \langle V, E \rangle$ *— совокупность двух множеств: непустого множества* V *(множества вершин) и множества* $E = \langle v \in V, q \in V \rangle$ *— двухэлементных подмножеств множества* V *(*$E-$ *множество ребер).*

Определение 2.7 *[19] Нагруженный граф − тройка* $< G, A, W >$*, где* G *— граф,* A *— алфавит атомов,* W *— функция нагрузки, со значениями из* A *и заданная на множествах вершин и ребер графа.*

Определение 2.8 *[38] Сепаратор двух вершин в нагруженном графе — пересечение нагрузок соответствующих вершин.*

Определение 2.9 *[38] Мощность сепаратора — число элементов в заданном сепараторе.*

Для удобства в данной работе понятия вершины v и нагрузки вершины W_v отождествляются.

Формальное определение графа смежности:

Определение 2.10 *[24] Неориентированный граф* $G = \langle V, E \rangle$ *— является графом смежности, если он удовлетворяет следующим условиям:*

1) $\forall u, v \in V$ *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:* $W_u \cap W_v \subseteq W_s$ *(магистральность);*

2) $\forall u, v \in E \quad W_u \cap W_v \neq \emptyset$*;*

3) $\nexists\ u, v \in V : W_u \subseteq W_v$*.*

Свойство магистральной связности является далеко немаловажным в теории АБС. При изменении одного из узлов именно это свойство обеспечивает корректность изменений всех остальных узлов. Изменения локализуются и далее могут быть обработаны локально от узла к узлу, не требуя глобального обновления всего графа. Данное свойство обеспечивает удобство вычислений.

Граф смежности с минимальным числом ребер называется *минимальным* графом смежности.

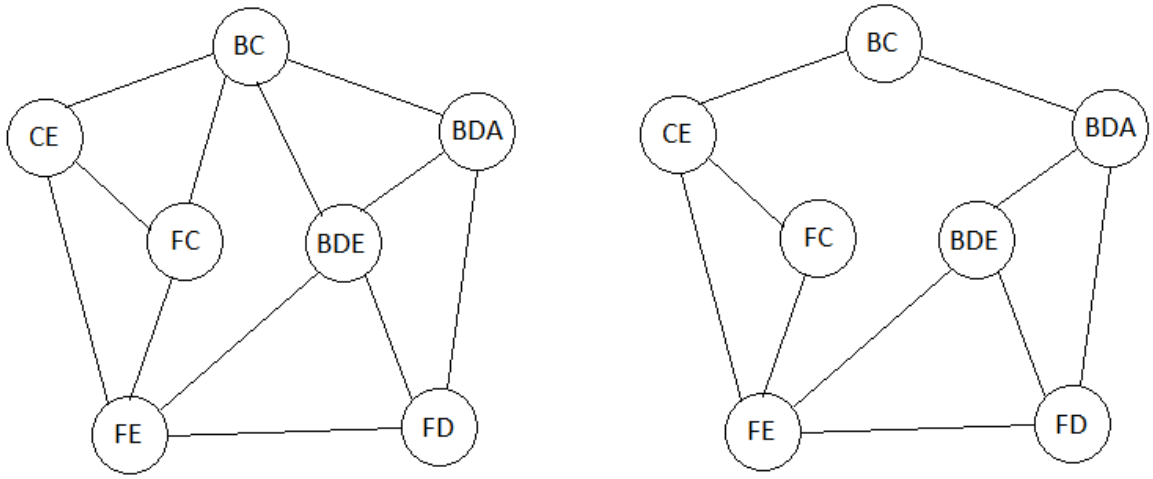

Рис. 5: Граф смежности и соответствующий ему минимальный граф смежности

2.3.2. Прямой алгоритм

Формализация данного алгоритма описана в [20], псевдокод представлен на листинге 1. Шаги данного алгоритма можно описать сле-

дующим образом:

- из заполненного в строках $4-7$ множества сепараторов извлекается очередной элемент;
- для каждой вершины далее проверяется, входит ли в ее нагрузку данный сепаратор — если да, то при пустом множестве S вычисляется множество достижимых вершин;
- далее, как указано в строках $14-16$, на каком-то шаге алгоритм соединит одну из вершин с текущей.

Algorithm 1 [20] Прямой алгоритм построения минимального графа смежности

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

```
1:  function STRAIGHT
2:      Q = ∅
3:      G =< V, ∅⟩
4:      foreach (u in V)
5:          foreach (v in (V \ u))
6:              if ((W_u ∩ W_v) ≠ ∅ && (W_u ∩ W_v) not in Q) then
7:                  Q = Q ∪ (W_u ∩ W_v)

8:      while (Q ≠ ∅)
9:          q = Q.Top()
10:         S = ∅

11:         foreach(v in V)
12:             if (q in Wv && (v not in S)) then
13:                 if (S ≠ ∅) then
14:                     d = Delegate(S)
15:                     S = S ∪ Component(G, v, q)
16:                     G.E = G.E ∪ (d, v)
17:                 else
18:                     S = Component(G, v, q)

19:     return G
20: end function
```

Обрабатывая каждый сепаратор из множества Q, алгоритм гарантированно соединит вершины, сепараторы которых не пусты. Отсутствие лишних ребер и тот факт, что получаемый в результате граф является минимальным графом смежности, обеспечивается проверкой в строке того, что текущая вершина, содержащая сепаратор q не входит в множество достижимых вершин, которые также содержат сепаратор q (строка 12 алгоритма).

2.3.3. Жадный алгоритм

Данный алгоритм отличается от предыдущего тем, что на первом шаге добавляет все возможные ребра, а затем удаляются те ребра, отсутствие которых не нарушит магистральную связность графа.

Можно выделить следующие шаги алгоритма:

- выбирается еще необработанное ребро;
- если нет необработанных ребер — построение графа завершается;
- если есть, то происходит проверка возможности удаления данного ребра без потери свойства магистральной связности;
- если такое возможно — ребро удаляется;
- в противном случае ребро помечается как обработанное.

Псевдокод данного алгоритма представлен на листинге 2. Более детально данный алгоритм представлен в [10].

2.3.4. Инкрементальный алгоритм добавления вершины в МГС

На вход алгоритму подается минимальный граф смежности и новая вершина. Необходимо построить минимальный граф смежности, множество вершин которого содержит новую добавленную вершину.

Стоит обратить внимание на тот факт, что в данном алгоритме отсутствует необходимость синтеза нового множества ребер для результирующего минимального графа смежности: инкрементальный алгоритм позволяет воспользоваться результатами предыдущих шагов.

Algorithm 2 [10] Жадный алгоритм построения минимального графа смежности

input: V, W **output**: $G = \langle V, E \rangle$

```
1:  function GREEDY
2:      G = ⟨V, ∅⟩
3:      foreach (u in V)
4:          foreach (v in (V \ u))
5:              if ((W_u ∩ W_v) ≠ ∅) then
6:                  G.E = G.E ∪ (u, v)

7:      while (true)
8:          edge = ∅
9:          foreach(e in G.E)
10:             if (e.RemoveAllowed) then
11:                 edge = e
12:                 break

13:         if (edge == ∅) then
14:             break

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

19:     return G
20: end function
```

В следующем листинге представлен схематичный алгоритм добавления новой вершины в МГС. На вход алгоритму подается МГС G и вершина v, на выходе – минимальный граф смежности G' содержащий новую вершину v.

Algorithm 3 [16] Инкрементальный алгоритм добавления 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

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

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

20:     return G'
21: end function
```

Алгоритм можно поделить на 2 части. В первой части (строки $2-7$) происходит добавление всех допустимых ребер во вторичную структуру АБС. Во второй части ($8-19$) осуществляется удаление ребер, без которых не нарушается магистральная связность вершин графа.

Данный алгоритм впервые формаизован и описан в [16].

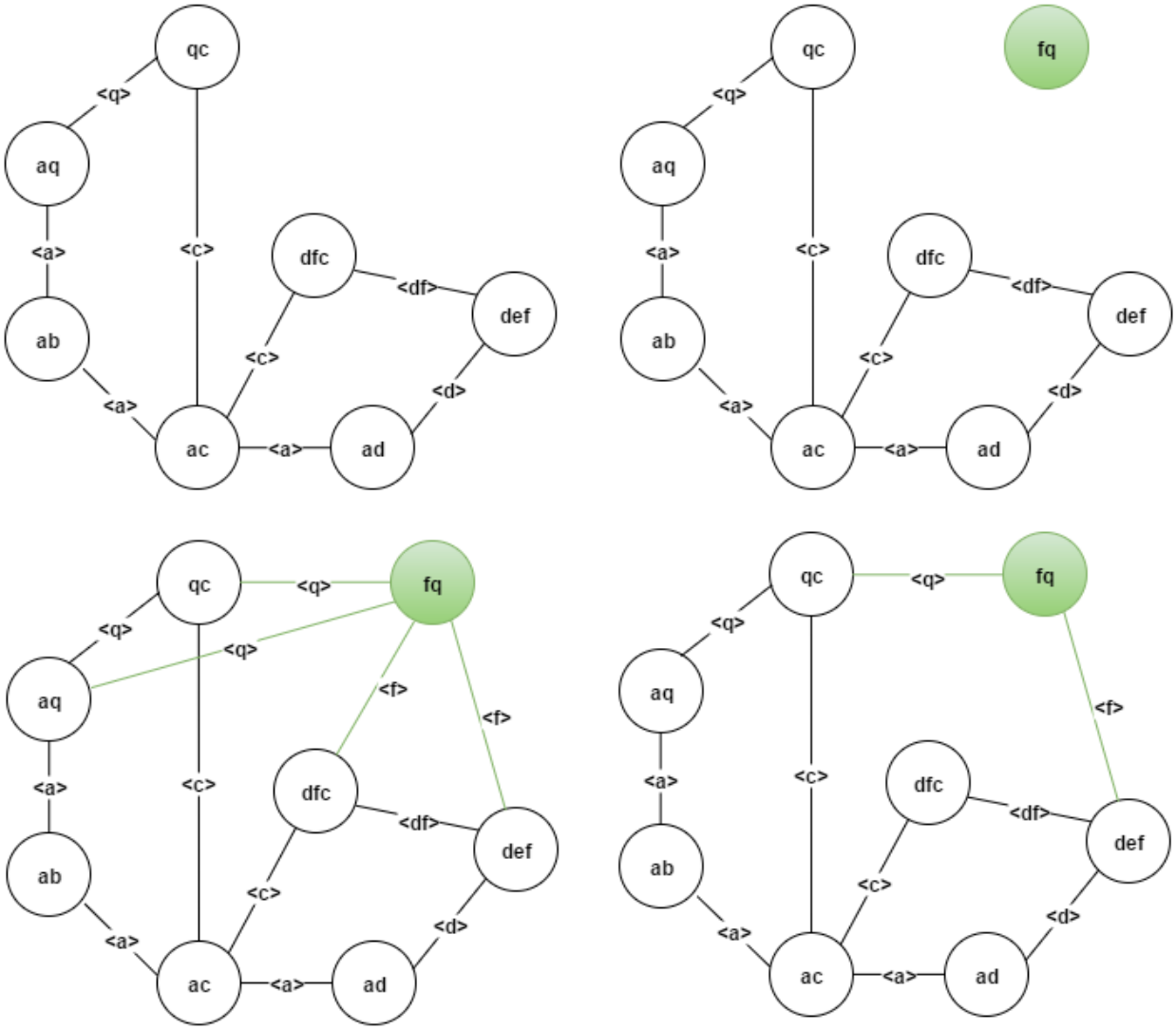

Рис. 6: [16] Пример добавления новой вершины в граф с сохранением свойства минимальности

2.3.5. Улучшенный инкрементальный алгоритм добавления вершины в минимальный граф смежности

На вход алгоритму поступает минимальный граф смежности и вершина, которую необходимо добавить в граф. Необходимо построить новый граф такой, что полученный граф будет являться графом смежности и множеством его вершин будет множество вершин исходного графа и новая вершина, полученная на входе.

Данный алгоритм отличается от инкрементального тем, что производится отбор ребер – кандидатов на добавление в граф. Добавляются

не все возможные ребра, а те из них, сепаратор которых удовлетворяют определенным условием.

Отбор ребер осуществляется с помощью алгоритма GetVerticesTo Connect, который проводит анализ вершин и их сепараторов и на основе этого анализа возвращает те ребра, добавление которых не будет избыточным. Подробнее описание данного алгоритма и его реализации приводится в [16, 17].

2.3.6. Декрементальный алгоритм удаления вершины из минимального графа смежности

На вход алгоритму подается минимальный граф смежности G и вершина v. Необходимо построить минимальный граф смежности таким образом, что множеством вершин полученного графа будет являться разностью исходного набора вершин и вершины v. Данный алгоритм представлен на листинге 4.

Кратко декрементальный алгоритм может быть описан следующими шагами:

- формируется граф смежности со множеством вершин $V \backslash v$;
- множество ребер будет состоять из множества ребер исходного графа без ребер, являющихся инцидентыми к вершине v и из всех возможных ребер между вершинами таких ребер;
- далее обрабатываются ребра и удаляются те, отсутствие которых не приведет к нарушению магистральной связности.

Подробно данный алгоритм рассмотрен в [47].

2.4. Программные технологии

Для реализации алгоритмов синтеза минимального графа смежности и алгоритмов для их сравнения используется язык программирования C# и платформа .NET framework. В качестве среды разработки была использована Microsoft Visual Studio 2015.

Algorithm 4 [47, 15] 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'' = ∅
5:      V'' = ∅
6:      foreach (e in E)
7:          if (e.First == v) then
8:              V'' = V'' ∪ {e.Second}
9:              E'' = E'' ∪ {e}
10:         if (e.Second == v) then
11:             V'' = V'' ∪ {e.First}
12:             E'' = E'' ∪ {e}

13:     foreach (x in V'')
14:         foreach (y in V'')
15:             if ((W_x ∩ W_y) ≠ ∅) then
16:                 E' = E' ∪ {x, y}

17:     E' = E' \ E''
18:     V' = V' \ {v}
19:     G' = ⟨V', E'⟩

20:     while (true)
21:         edge = ∅
22:         foreach(e in E')
23:             if (e.RemoveAllowed) then
24:                 edge = e
25:                 break foreach

26:         if (edge == ∅) then
27:             break while
28:         if (PathExists(G', edge, edge.First, edge.Second)) then
29:             E' = E' \ {edge}
30:         else
31:             edge.RemoveAllowed = false
32:     return G' = ⟨V', E'⟩
33: end function
```

Для реализации frontend-части программного комплекса были использованы следующие технологии:

- язык TypeScript;
- библиотека d3.js;
- framework Angular v2;
- VS code.

На сегодняшний день JavaScript является кроссплатформенным языком, доступный практически для любых устройств. Однако JavaScript обладает рядом недостатков, затрудняющих написание кода и его отладку. К таковым относятся, например, следующие:

- динамическая типизация;
- отсутсвие модульности.

TypeScript, разработанный компанией Microsoft [51], является надмножеством языка JavaScript. TypeScript использует статическую типизацию, что означает проверку типов на этапе компиляции. TypeScript добавляет возможность объявлять модули, классы, интерфейсы. Код, написанный на TypeScript компиллируется в JavaScript-код, который подается на исполнение. Ввиду вышеуказанных преимуществ для имплементации frontend-составляющей программного комплекса был выбран именно TypeScript.

Библиотека *D3.js* (или просто D3: Data-Driven Documents или документы, управляемые данными)) – JavaScript–библиотека для обработки и визуализации данных, которая использует данные для создания и контроля интерактивных графических элементов, отображаемых в веб-браузере [45]. Более того, D3 как инструмент визуализации данных поддерживает технологии, предусмотренные стандартом W3C–SVG (ScalableVectorGraphics – масштабируемая векторная графика), JavaScript, HTML5 и CSS3. Стоит отметить, что технология

масштабируемой векторной графики позволяет достигнуть наилучшего качества изображения при любом разрешении.

При выборе данной библиотеки учитывались следующие ее преимущества:

1. Бесшовная интеграция с самыми современными технологиями — например HTML5 и CSS3.

2. Использование стандартной, универсальной модели DOM, которая не только работает с любым языком, но и позволяет представить любой документ (в частном случае, представление набора данных) в удобном для конечного пользователя формате.

3. Прозрачность разработки — функции библиотеки не скрыты внутри неких классов (как это происходило с некоторыми ее предшественниками), а могут использоваться в любом месте документа при условии обращения к библиотеке.

4. Контроль над представлением данных — как в рамках программирования, так и относительно конечного результата визуализации.

AngularJS v.2 это open - source набор библиотек, который предназначен по большей части для разработки одностраничных приложений. Указанный фреймворк был разработан компанией Google, также разработавшей предыдущую версию фреймворка — AngularJS [43]. Angular2 написан на языке программирования TypeScript, преимущества которого были рассмотрены выше, и поддерживает написание кода, использующего фреймворк, на таких языках программирования, как TypeScript, Dart [46] и Java Script.

В качестве редактора кода использовался VS code, используемая система контроля версий — git. Стоит отметить, что для хостинга проектов использовался веб-сервис BitBucket, основанный на системе контроля версий Mercurial и Git [44]. По назначению и предлагаемым функциям аналогичен GitHub, однако предоставляет бесплатные закрытые

репозитории. Данный сервис также предоставляет возможности отслеживания выполнения поставленных задач, а также проведения ревью программного кода, что улучшает и упрощает совместную разработку — данные преимущества являются немаловажными для выполнения проектной работы.

Таблица 1: Используемые технологии и тип лицензии

Технология	Тип лицензии
d3.js	BSD
AngularJS v2	MIT
TypeScript	OWFa 1.0, Лицензия Apache (компилятор)
VS code	MIT License (при сборке из исходных кодов), Microsoft Pre-Release Software License (для готовых сборок)

2.5. Выводы к главе

В данной главе введены основные понятия теории АБС: фрагмент знаний, первичная и вторичная структуры, которые будут использоваться далее; приведен псевдокод алгоритмов генерации минимального графа смежности. Также рассмотрены программные технологии, используемые в данной работе.

3. Алгоритмы синтеза МГС: сравнительный анализ

3.1. Введение

В данной главе рассматриваются подход и вспомогательные инструменты, которые были использованы для проведения сравнения алгоритмов синтеза минимального графа смежности. Также в данной главе описываются проведенные эксперименты и полученные результаты.

3.2. Описание вычислительных экспериментов

Алгоритмы генерации минимального графа смежности были реализованы на языке $C\#$. Соответственно на языке $C\#$ был написан как сам алгоритм проведения вычислительных экспериментов, так и алгоритм генерации тестового набора данных.

Для сведения к минимуму влияния таких факторов, как состояние системы, на которой запускается алгоритм, каждый набор данных передается алгоритму k раз ($k = 50$). Затем рассчитывается среднее значение времени обработки входных данных. Таким образом, получается среднее время работы алгоритма. Так как сам набор данных также влияет на время работы алгоритма, формируется ряд наборов ($m = 60$ наборов), обладающих одинаковыми свойствами, а именно одинаковым количеством вершин и одинаковым распределение мощностей нагрузок.

Для каждого набора данных для сравниваемых алгоритмов вычисляется отношения времен их работы и натуральный логарифм отношения времен работы. Затем, для сравнения двух алгоритмов вычисляются необходимые статистики.

Условно, существующие алгоритмы синтеза можно поделить на 2 типа: алгоритмы, синтезирующие граф заново (к таким относятся прямой и жадный алгоритмы) и алгоритмы, модифицирующие уже существующие структуры (инкрементальный, декрементальный и улучшенный инкрементальный). Таким образом и подходы к проведению экспери-

ментов по сравнению работ алгоритмов должны быть различными для разных сравнений. Ранее были проведены работы по сравнению прямого и жадного алгоритмов [13], а также описана общая методика сравнения [11]. Также, для проведения сравнения алгоритмов, относящихся к разным типам, решения были предложены в [8, 10, 13]. Для сравнения инкрементального и улучшенного инкрементального, а также инкрементального и декрементального необходимо создать собственную модификацию алгоритма для проведения вычислительных экспериментов.

Таким образом, были созданы модифицированные версии алгоритма *Experimental*, представленного в [8, 10, 12], а именно $ExpreimentIncr\ Incr$ и $ExpreimentIncrDecr$. Используемый алгоритм представлен в листинге 5.

В каждую итерацию внешнего цикла создается множество вершин и вершина, которую необходимо добавить в граф. Метод *generateGraph* создает минимальный граф смежности, используя жадный алгоритм. Далее во внутреннем цикле (строки 12-13) замеряется время работы алгоритмов, на вход которым поступает минимальный граф смежности и вершина, которую необходимо добавить. Алгоритм возвращает отсортированный набор отношений времен работы двух алгоритмов.

Для получения данных сравнения работы инкрементального и декрементального алгоритмов, использовался алгоритм, схожий с алгоритмом, представленном на листинге 5, за исключением некоторых модификаций, а именно:

- дополнительно генерируется граф $G' = generateGraph(\langle V \cup v \rangle, \emptyset)$;
- на вход инкрементальному алгоритму подается граф G, множество вершин которого не содержит v, и вершина v: $Incr(G, v)$;
- на вход декрементальному алгоритму подается граф G и вершина v, содержащаяся во множестве вершин графа.

Algorithm 5 Алгоритм оценки времени работы инкрементального и улучшенного инкрементального алгоритмов

input: N, $alhabet$, $distributions$, $distribution$, $Incr$, $ImprovedIncr$
output: $ratios$

```
1:  function ExperimentalIncrIncr
2:      m = 60
3:      timeRatios = ∅
4:      for (i = 0; i < m; i++)
5:          timeForIncr = 0
6:          timeForImprovedIncr = 0
7:          k = 40
8:          v = TestSets(1, alphabet, distribution)
9:          V = TestSets(N, alphabet, distributions)
10:         G = generateGraph(V, ∅)

11:         for (j = 0; j < k; j++)
12:             timeForIncr+ = getTime(Incr(G, v))
13:             timeForImprovedIncr+ =
14:                     getTime(ImprovedIncr(G, v))

15:         averageForImprovedIncr = timeForImprovedIncr/k
16:         averageForIncr = timeForIncr/k
17:         ratioOfTimes.Add(averageForImprovedIncr/averageForIncr)

18:     return ratioOfTimes.Sort()
19: end function
```

3.3. Результаты вычислительных экспериментов

Результаты сравнения алгоритмов

Для того, чтобы более точно охарактеризовать полученные значения логарифмов отношений времен работы алгоритмов использовались следующие показатели:

- среднее R_i;
- первый квартиль Q_1;
- третий квартиль Q_3;
- первый дециль d_1;
- девятый дециль d_9.

Подробнее подход к сравнению алгоритмов и выбору данных статистик рассмотрен в [6, 10, 13].

Поведение алгоритмов на выборках в среднем характеризуется средним. Для выявления хороших выборок используются 1 и 3 квартиль, а 1 и 9 — для выявления плохих выборок.

Использование квартилей и децилей необходимо потому, что с помощью данных показателей можно найти выбросы, то есть такие наборы данных, для которых время работы алгоритмов отличается от среднего значительно. Информация такого рода позволяет выявить наиболее оптимальный алгоритм для использования на конкретном наборе данных.

Также важную роль играют части графика, лежащие между 1 и 3 квартилем и между 1 и 9 децилем. Чем меньше размах, тем меньше вероятность наличия в выборке таких элементов, для которых работа алгоритмов значительно отклоняется от среднего.

В процессе проведения вычислительных экспериментов формируются различные наборы данных ($m = 60$ раз). Каждый из этих наборов подается на вход алгоритмам k раз ($k = 50$). Затем временные

значения, полученные на одном и том же наборе данных усредняются. Далее составляется набор отношений времен двух сравниваемых алгоритмов, полученных на разных наборах данных, которые, однако, являются одинаковыми по мощности.

Данный алгоритм проводился для различного набора мощностей. На графике представлены следующие статистики: Rg – среднее, первый и девятый децили D_1 и D_9, первый и третий квартили Q_1 и Q_3, MIN и MAX – минимальное и максимальное значения.

На рисунках 7 - 10 представлены графики вышеуказанные статистик относительного времени выполнения инкрементального и улучшенного инкрементального алгоритмов. На графике показано отношение времени работы инкрементального алгоритма ко времени работы улучшенного инкрементального алгоритма. Данный результат позволяет сделать вывод о том, что модификации, примененные к улучшенному инкрементальному алгоритму позволяют получить значительное ускорение работы алгоритма. Более явно улучшения видны на диапазоне $70 - 80$ количества вершин.

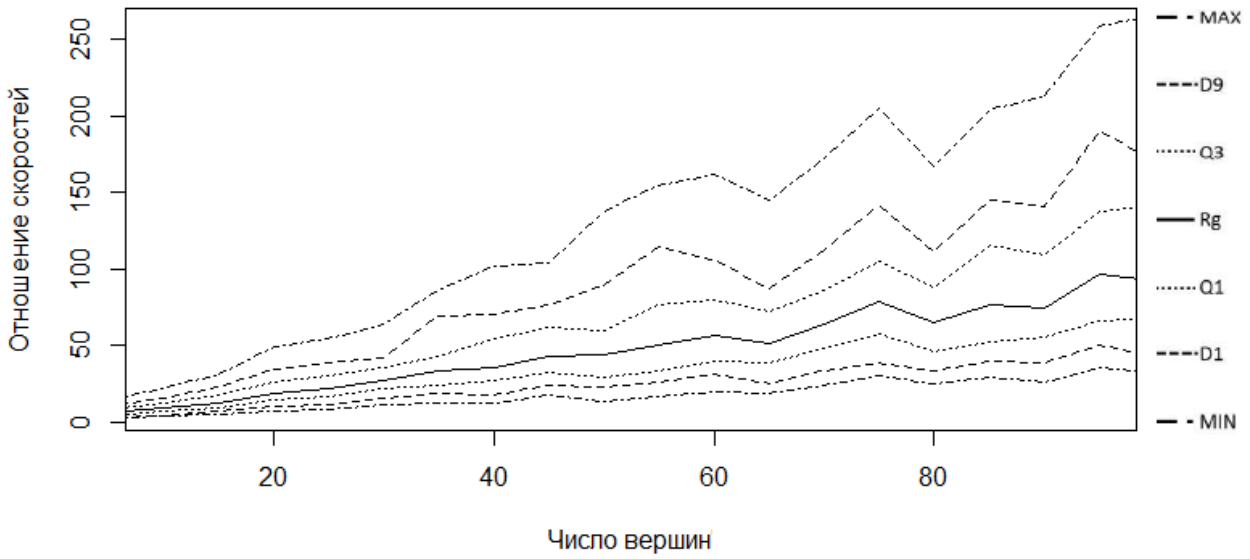

Рис. 7: Алфавит 61 символ. 88% вершин $2-4$ порядков, 12% – $5-7$ порядков, 0% – $8-10$ порядков, 0% – $11-13$ порядков

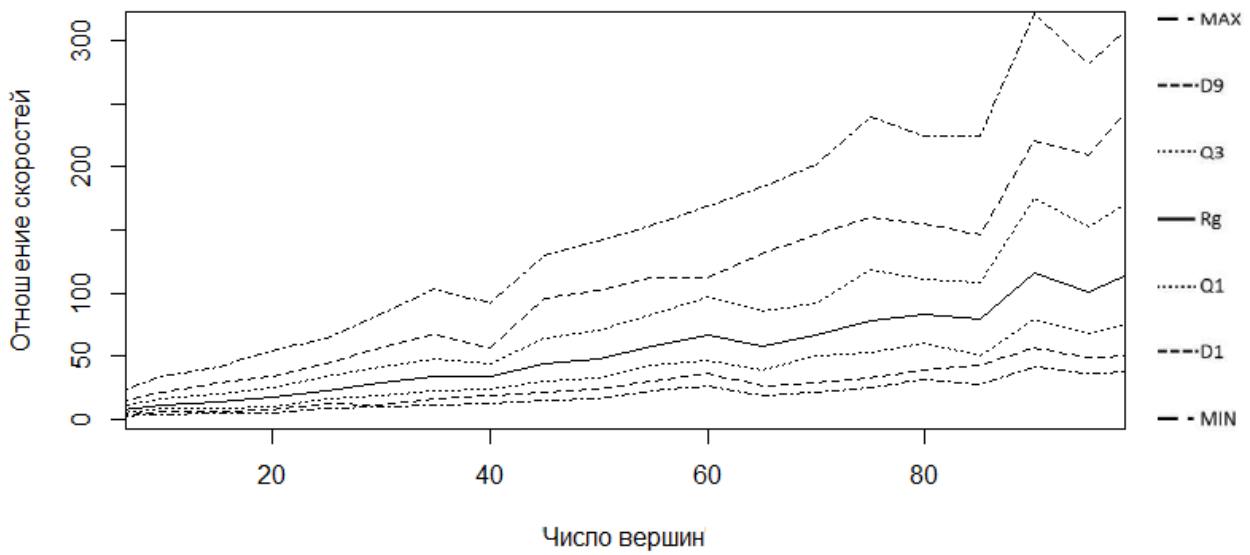

Рис. 8: Алфавит 61 символ.60% вершин $2-4$ порядков, $30\% - 5-7$ порядков, $6\% - 8-10$ порядков, $4\% - 11-13$ порядков

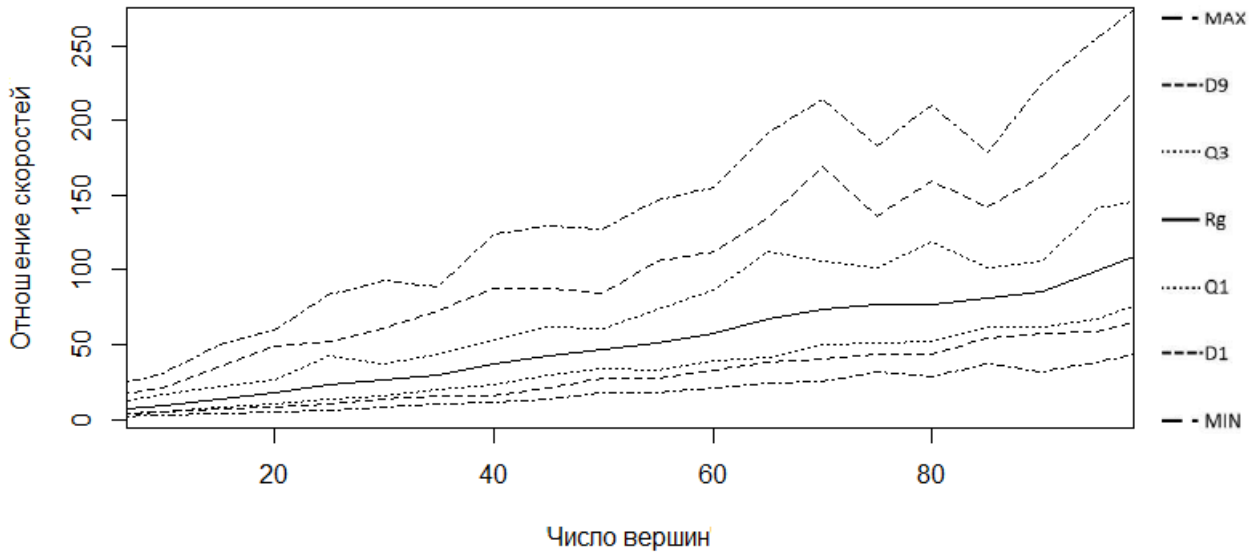

Рис. 9: Алфавит 61 символ. 48% вершин $- 2-4$ порядков, $30\% - 5-7$ порядков, $12\% - 8-10$ порядков, $10\% - 11-13$ порядков

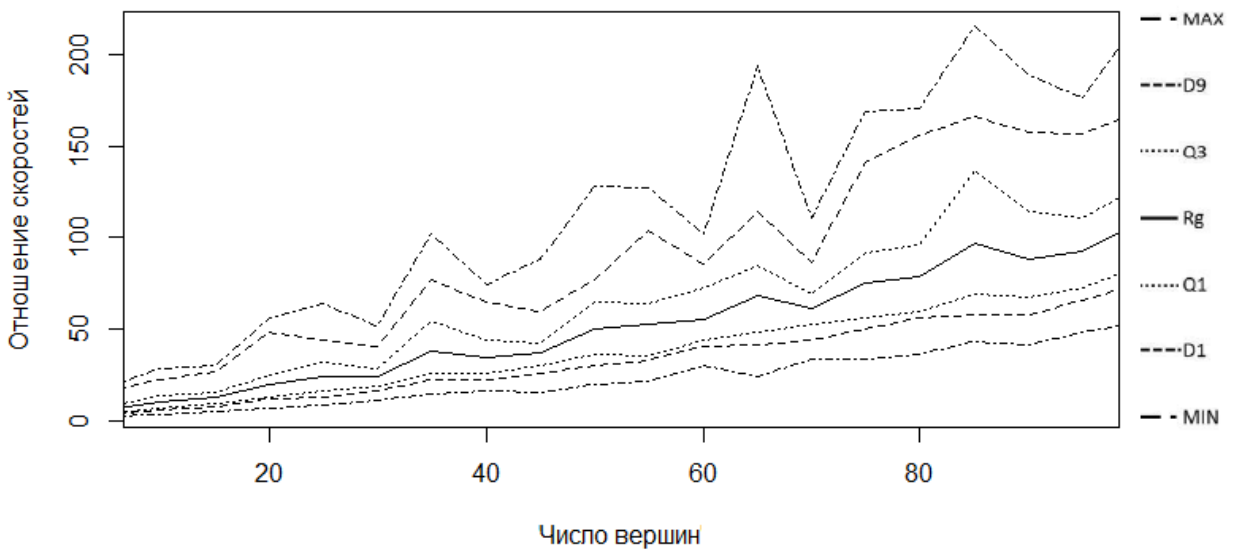

Рис. 10: Алфавит 61 символ. 18% вершин $2-4$ порядков, 32% $-$ $5-7$ порядков, 22% $-$ $8-10$ порядков, 28% $-$ $11-13$ порядков

На рисунках 11 - 14 представлен графики статистик сравнения инкрементального и декрементального алгоритмов синтеза минимального графа смежности. На данном графике рассматриваются отношения времени работы инкрементального алгоритма к декрементальному. График позволяет сделать вывод о том, что декрементальный алгоритм работает в среднем в $10-15$ раз быстрее инкрементального. Причем, значение отношения времен показывает небольшой рост с увеличением числа вершин. Можно сделать вывод о том, что операция удаления вершины с использованием данной реализации декрементальной вершины менее затратна по времени, чем добавление вершины с использованием инкрементального алгоритма.

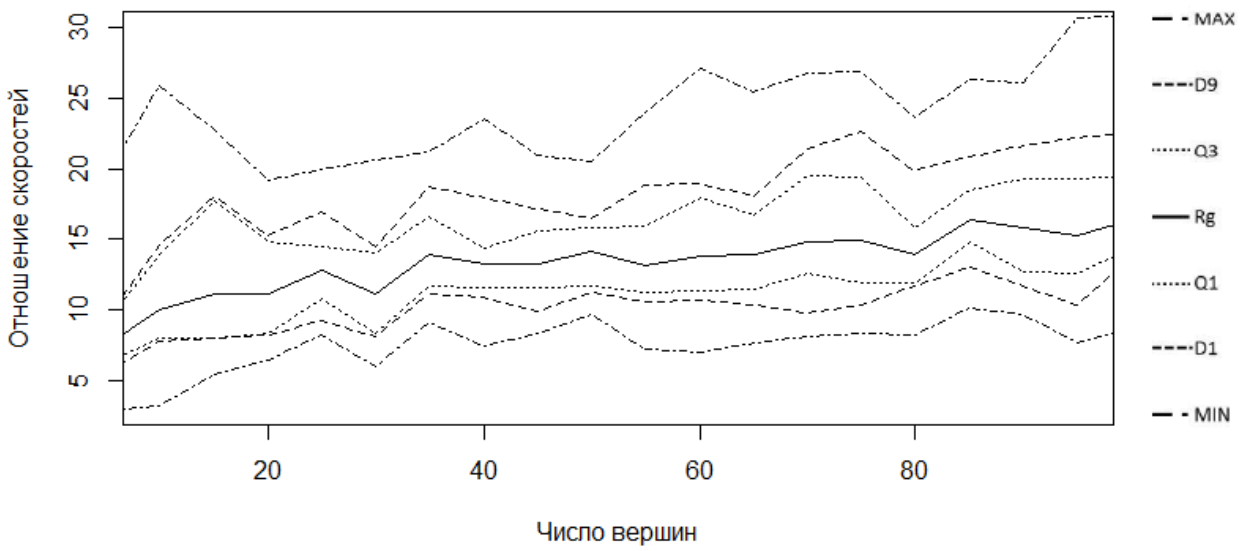

Рис. 11: Алфавит 61 символ. 88% вершин $2-4$ порядков, 12% $-$ $5-7$ порядков, 0% $-$ $8-10$ порядков, 0% $-$ $11-13$ порядков

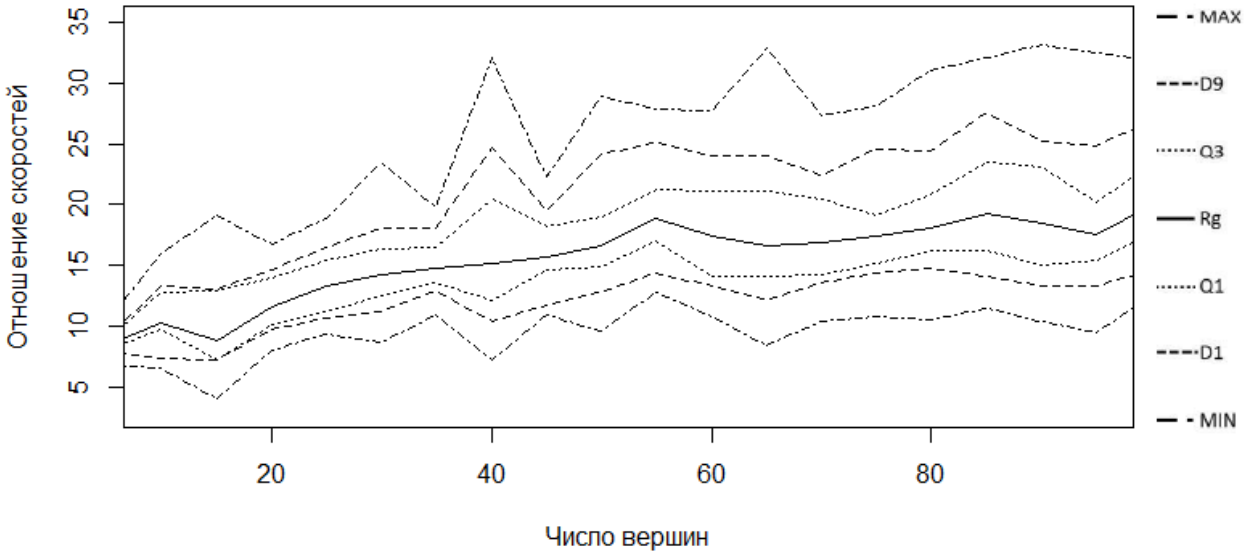

Рис. 12: Алфавит 61 символ.60% вершин $2-4$ порядков, 30% $-$ $5-7$ порядков, 6% $-$ $8-10$ порядков, 4% $-$ $11-13$ порядков

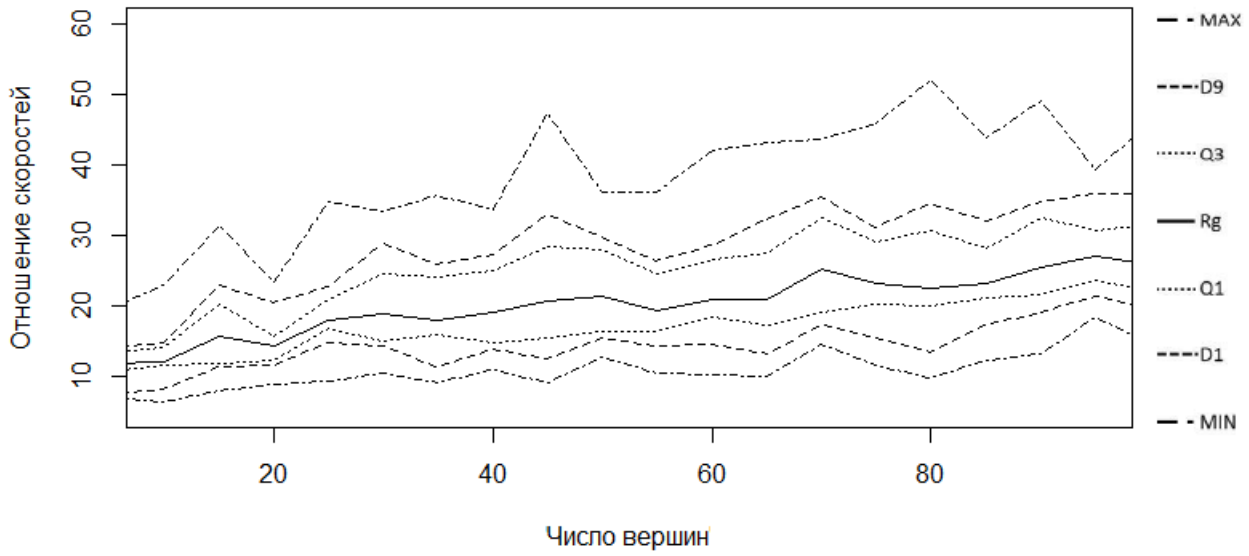

Рис. 13: Алфавит 61 символ. 48% вершин $-$ $2-4$ порядков, 30% $-$ $5-7$ порядков, 12% $-$ $8-10$ порядков, 10% $-$ $11-13$ порядков

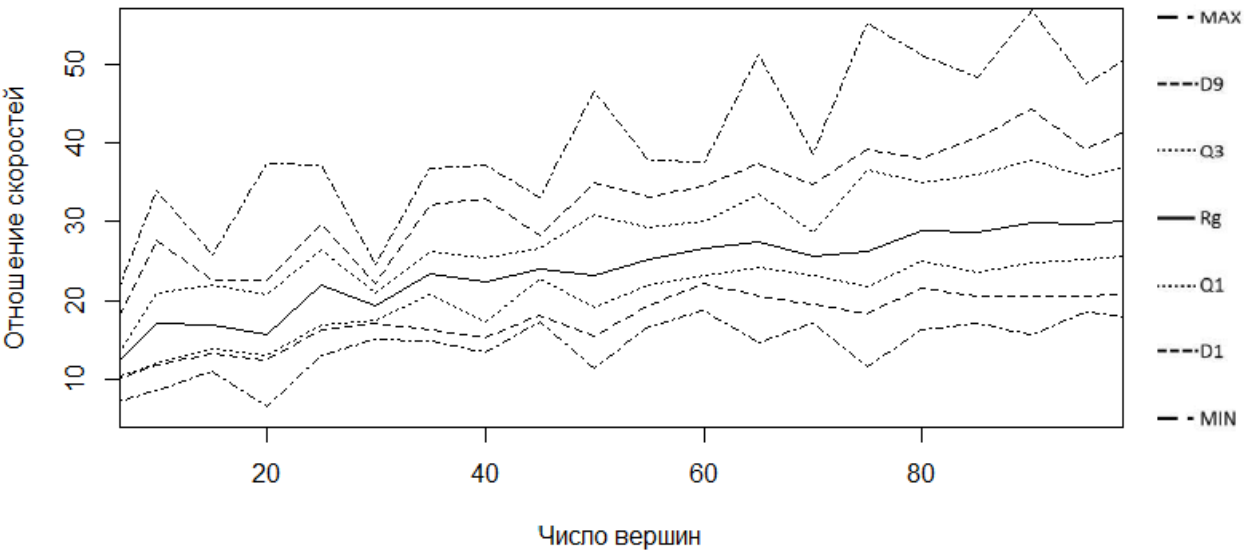

Рис. 14: Алфавит 61 символ. 18% вершин $2-4$ порядков, 32% $-$ $5-7$ порядков, 22% $-$ $8-10$ порядков, 28% $-$ $11-13$ порядков

3.4. Проверка данных на нормальность

Как было сказано ранее, при вычислении таких статистик, как верхняя и нижняя границы доверительного интервала, делалось предпо-

ложение, что, после логарифмирования, выходные данные алгоритма подчиняются закону нормального распределению. Для того, чтобы убедиться в этом, было предложено использовать критерии проверки данных на нормальность. Кроме того, логарифмированные данные можно визуализировать в виде гистограммы, дополнительно указав кривую эмпирического и теоретического распределений.

С помощью возможностей языка R и визуальной среды разработки RStudio [49, 48] были получены гистограммы для выходных данных алгоритмов сравнения. Для проверки гипотезы о нормальности полученных данных использовался критерий Шапиро-Уилка [14]. Результаты проверки гипотезы указаны на графиках.

Вид кривых и распределение данных, показанное на рисунках позволяет сделать вывод о том, что распределение данных имеет вид, схожий с нормальным. Если также учесть результаты критерия, то можно сделать вывод о том, что предположение о нормальности распределения логарифмов отношений времен работы алгоритмов, верно.

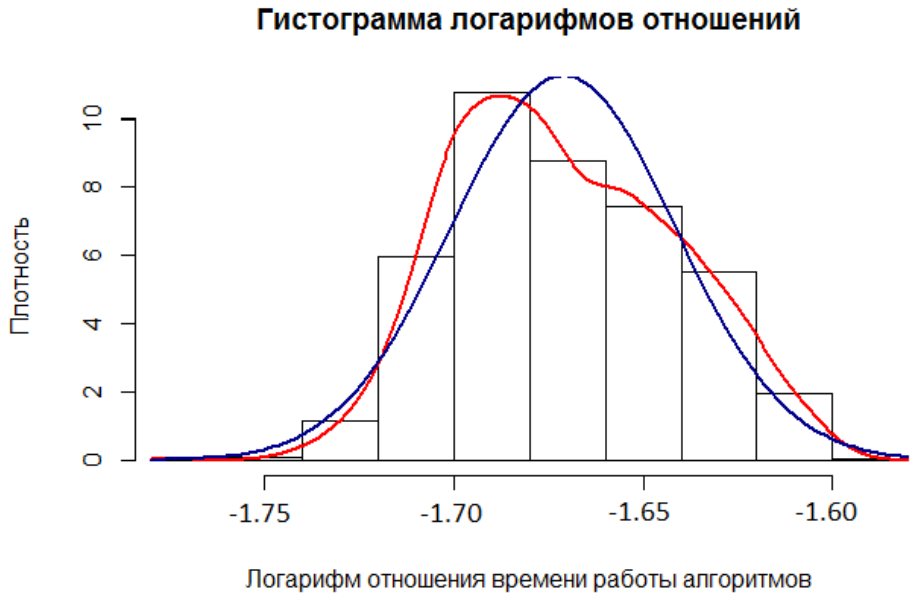

Рис. 15: Гистограмма распределения логарифмов отношения времен работ инкрементального и улучшенного инкрементального алгоритмов

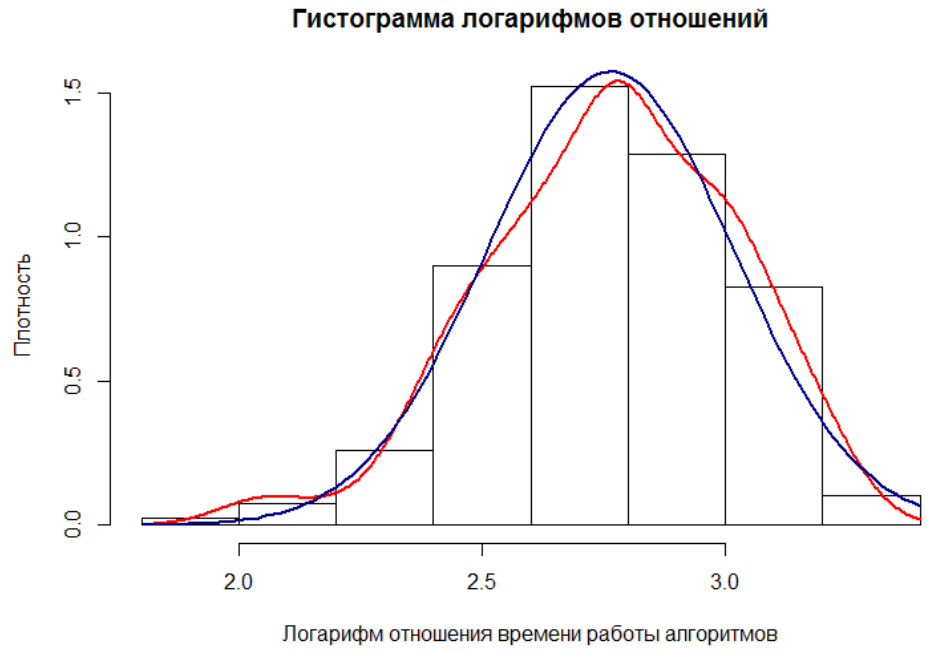

Рис. 16: Гистограмма распределения логарифмов отношения времен работ инкрементального и декрементального алгоритмов

3.5. Выводы к главе

В данной главе описан примененный подход к сравнению алгоритмов, приведены результаты. Сравнение инкрементального и улучшенного инкрементального алгоритма показало, что примененные к алгоритму модификации существенно ускоряют процесс синтеза. Проверка на нормальность показала, что логарифмы отношений времен работы алгоритмов распределены по закону, близкому к нормальному, а значит имеет место использование статистических показателей, указанных ранее.

4. Визуализация структур АБС

4.1. Введение

В данной главе представлен пользовательский интерфейс разработанного комплекса программ, приведен интерфейс созданных классов и некоторые фрагменты кода, а также освещены возможности, предоставляемые данным комплексом.

На рисунке 17 представлена структура разрабатываемого веб - приложения.

Рис. 17: [5] Диаграмма компонент проекта

Представленные компоненты являются обособленными и для каждой их них была реализована собственная логика работы. В рамках данной работы была реализована клиентская часть веб-приложения.

Каждый проект состоит из списка АБС, для каждой из которых предоставлены возможности редактирования и обработки пользователем независимо от других пользователей. Каждый проект имеет название и описание. Каждая алгебраическая байесовская сеть состоит из множества ФЗ, образующих первичную структуру, и множества вторичных структур. Пользователю доступны возможности создания и редактирования структур, а также осуществление всех видов логико-вероятностного вывода.

4.2. Визуализация компонентов АБС

Известным является тот факт, что в процессе обучения, информация, подкрепленная визуальным представлением, усваивается лучше. Следовательно, для пользователя системы, а именно обучающегося или исследователя, важна возможность наглядного представления материала и интерактивное взаимодействие со структурами.

Основными объектами, над которыми производятся манипуляции в алгоритмах логико — вероятностного вывода и инкрементально — декрементальных алгоритмах, являются локальная структура фрагмента знаний и глобальная структура АБС — именно эти структуры визуализируются в веб-приложении.

4.2.1. Описание общей структуры класса

Для создания визуального представления графов используется ранее упомянутая библиотека D3.js. В данной библиотеке функции API разделены на логические блоки. Одним из таких логических блоков является блок Layouts. Функционал, необходимый для визуализации относительно друг друга элементов, которые связываются с данными. Layouts проводит обработку данных последующей визуализации.

Разновидностью Layouts является Force Layout, который представляет собой ни что иное, как реализацию алгоритмов визуализации графов на основе сил (Force-directed graph drawing algorithms). С помощью данного класса алгоритмов проводится вычисление позиции каждого из узлов на основе моделирования силы притяжения между каждой парой связанных узлов, а также отталкивающую связь между ними. Подробная информация о физическом моделировании представлена в [50]. Важным отличаем Force Layouts от других Layouts является тот факт, что в качестве узлов могут быть использованы различные сущности, а не только circle. Такими сущностями могут быть текст или изображение.

В целом, как первичная, так и вторичная структура представляют собой граф, поэтому целесообразным представляется использование об-

щего абстрактного класса, содержащего необходимые для обеих структур методы. Интерфейс такого класса представлен далее на листинге 1.

Listing 1 Интерфейс абстрактного класса

```
export class AbstractChart {
    constructor(initialConfig: IAbnChartConfig,
                helper: AbnChartHelper){}

    public draw() {}
    protected getDefaultConfig() {}
    protected getPreparedNodes() {}
    public redraw() {}
    public cleanChart(){}
}
```

На вход конструктору класса предоставляется параметр initialConfig, содержащий данные о размере окна и полученные данные о вершинах и дугах графа. Основным и самым важным методом является метод draw(), который производит отрисовку необходимых элементов. Также класс содержит методы очистки графика и перерисовки (cleanChart() и redraw() соответственно). Так как первичная и вторичная структуры содержат отличающиеся друг от друга интерфейсы предоставляемых данных, методы абстрактного класса getPreparedNodes() и getDefault Config() должны быть прописаны для каждого из этих классов отдельно.

Далее на листинге 2 представлен интерфейс IAbnChartConfig, содержащий необходимую информацию для корректного построения графика, и интерфейс класса AbnChartHelper содержащего вспомогательные функции.

Listing 2

```
export interface IAbnChartConfig {
    width: number;
    height: number;
    data: IAbnChartData;
    alphabet: string[];
}
```

Поля *width* и *height* содержат ширину и высоту соответственно отведенной под график области. Поле *alphabet* содержит набор строк представляющих собой псевдонимы для атомов. Поле *data* содержит в себе данные, а именно сведения о дугах и вершинах графа.

Listing 3

```
export class AbnChartHelper {
    combinations(){}
    intersection() {}
    isConnected() {}
    generateTestFirstly() {}
    generateTestSecondary() {}
}
```

Класс AbnChartHelper содержит вспомогательные функции необходимые для обработки данных, а также для генерации тестовых наборов.

4.2.2. Первичная структура

В роли первичной структуры в теории АБС выступает набор фрагментов знаний. Так как фрагмент знаний имеет устоявшееся визальное представление, то было решено использовать именно его.

Как было сказано ранее, фрагмент знаний представляет собой идеал конъюнктов. Устоявшееся визуальное представление фрагмента знаний это расположение конъюнктов по уровням в зависимости от количества

членов, содержащихся в конъюнкции. На практике редко используются фрагменты знаний, содержащие в себе больше трех атомов.

На вход подается набор атомов. Создаются все возможные конъюнкции и далее распределяются по уровням в зависимости от количество членов конъюнкции. Класс AbnLocalStructureChart является наследником класса AbstractChart. Класс содержит в себе переопределенные методы generateDefaultConfig и getPreparedNodes. Метод getPrepared Nodes с помощью вспомогательных методов высчитывает позицию каждого узла и преобразует данные. Интерфейс класса представлен на листинге 4.

Listing 4

```
export class AbnLocalStructureChart extends AbstractChart{
    protected generateDefaultConfig() {}
    protected getPreparedNodes() {}
}
```

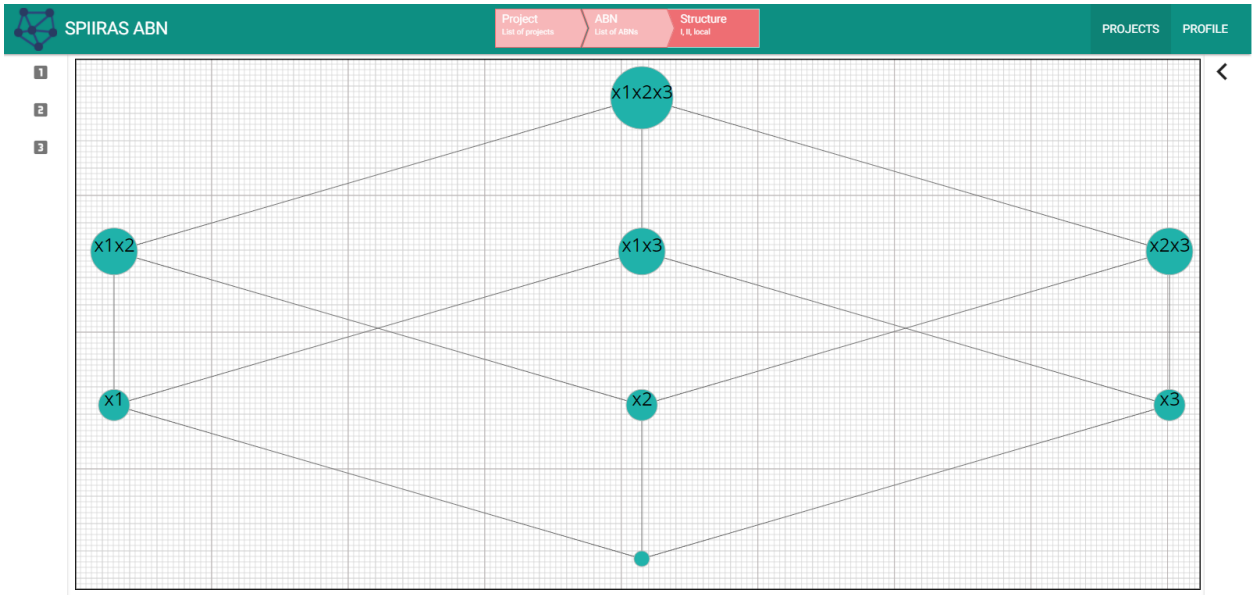

Рис. 18: Визуальное представление первичной структуры в комплексе программ

4.2.3. Вторичная структура

Так как для вторичной структуры не существует некоторого устоявшегося в литературе представления было решено визуализировать данную структуру следующим образом.

Вершины выстраиваются по уровням таким образом, что вершина с наибольшей степенью (количеством инцидентных ребер) располагалась в левой части выделенной под граф области. Реализация данного подхода предполагает поиск вершины с наибольшей степенью и дальнейшее распределение узлов по уровням: на первом уровне располагается вершина с наибольшей степенью, затем на втором уровне — вершины, смежные с вершинами предыдущего уровня. Далее построение осуществляется рекурсивно. После распределения вершин по уровням, необходимо высчитать координаты каждого из узлов в зависимости от количества уровней, количества вершин на одном уровне с обрабатываемым, а также в зависимости от ширины и высоты отведенной под граф области.

Класс AbnSecondaryStructureChart имеет такой же интерфейс, как и AbnLocalStructureChart, однако содержит некоторые дополнительные методы.

Listing 5 Интерфейс вторичной структуры

```
1 export class AbnSecondaryStructureChart extends AbstractChart{
2     protected generateDefaultConfig() {}
3     protected getPreparedNodes() {}
4 }
```

4.3. Примеры работы

Для визуализации структуры был использован упомянутый выше блок Force Layout библиотеки D3.js. Данный блок также предоставляет возможности интерактивного взаимодействия с графом, а именно возможность перемещения вершин, возможность обработки ряда раз-

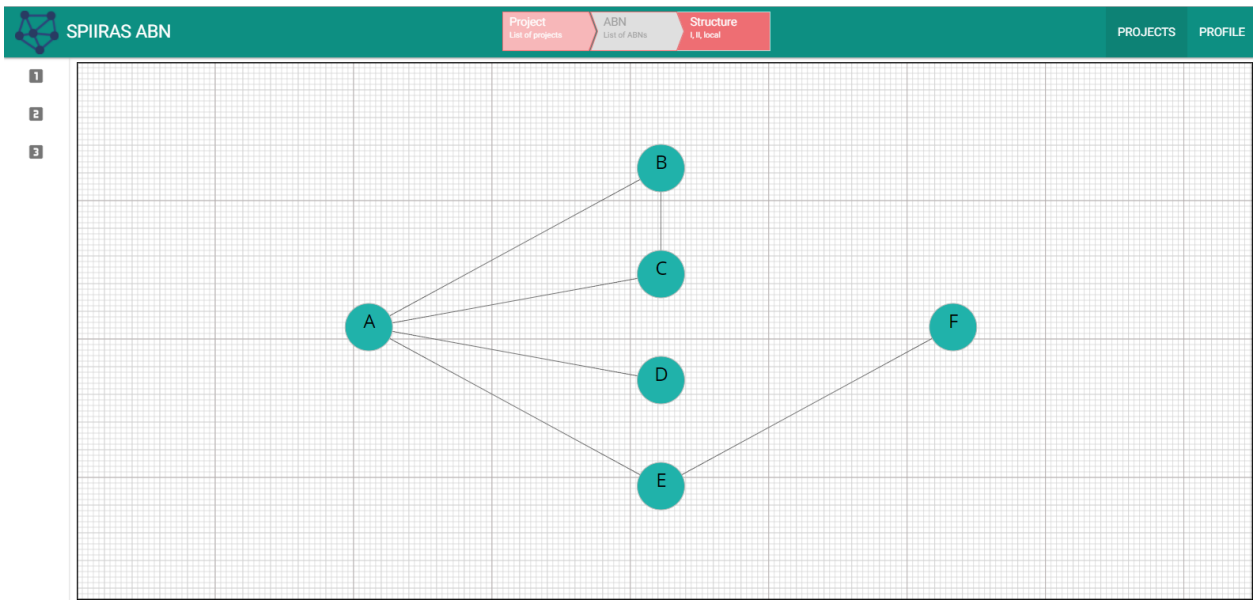

Рис. 19: Визуальное представление вторичной структуры в комплексе программ

личных событий (click, hover) над визуальными элементами. В процессе реализации подхода к визуализации фрагмента знаний возникла необходимость сохранять положения элементов графика — это необходимо при перемещении элементов пользователем, а также при изменении размера окна. Также, полезной является функция подсветки смежных узлов графа. Данные возможности были успешно реализованы.

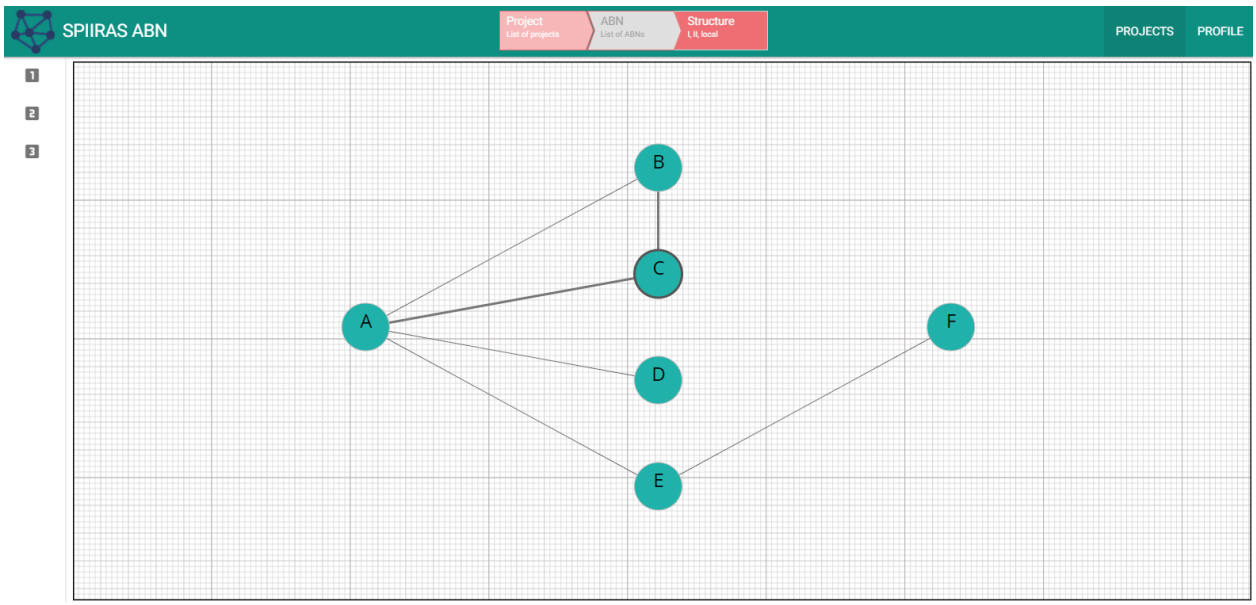

Рис. 20: Интерактивные возможности: выделение смежных вершин и инцидентных дуг

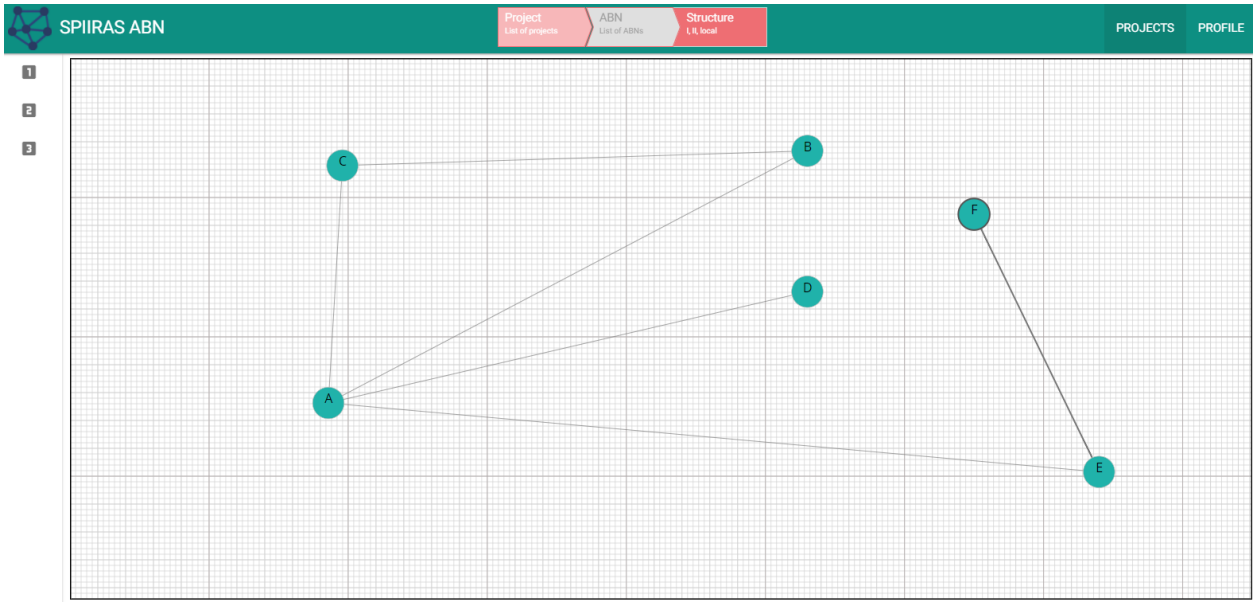

Рис. 21: Интерактивные возможности: перемещение вершин с возможностью сохранения конфигурации

4.4. Выводы к главе

В данной главе была рассмотрена реализация визуального представления компонент алгебраической байесовской сети, а также интеграция frontend-составляющей с другими частями веб-приложения. На данный момент программный комплекс обладает основной функциональностью, а именно: возможности создания фрагмента знаний и вторичной структуры, а также возможность осуществление логико-вероятностного вывода.

Заключение

В ходе выполнения данной работы была достигнута поставленная цель, а именно: визуализация глобальных структур и автоматизация анализа алгоритмов их синтеза. Поставленные для достижения цели задачи были выполнены. Получены следующие результаты:

1. получены результаты вычислительных экспериментов над конкурирующими алгоритмами синтеза МГС;
2. обоснована корректность вычислительных экспериментов;
3. разработаны алгоритмы визуализации структур АБС;
4. обеспечена интеграция frontend-составляющей с другими частями системы;
5. программный комплекс протестирован, создан ряд примеров, демонстрирующих ее работу.

Анализ алгоритмов синтеза минимального графа смежности, а именно инкрементального, улучшенного инкрементального и декрементального, позволил сравнить временные характеристики алгоритмов. Это предоставляет основу для анализа применимости данных алгоритмов, а также для возможных подходов к их усовершенствованию.

Реализация программного комплекса в виде веб-приложения открывает новые возможности для использования аппарата алгебраических байесовских сетей и его популяризации. Данное решение, в условиях наличия стабильного интернет-соединения, позволяет быстро получить доступ ко всем возможностям системы, что упрощает организацию учебной деятельности, связанной с алгебраическими байесовскими сетями.

Результаты данной работы вошли в следующие публикации:

1. Березин А.И., Зотов М.А., Иванова А.В. Синтез множества минимальных графов смежности: статистическая оценка сложности

инкрементального алгоритма // СПИСОК-2016: Материалы всероссийской научной конференции по проблемам информатики. Санкт - Петербург, 2016. С. 443–453.

2. Березин А.И., Иванова А.В., Зотов М.А. Синтез множества минимальных графов смежности: статистическая оценка сложности декрементального алгоритма // Международная конференция по мягким вычислениям и измерениям. 2016. Вып. Секции 1-3. Т. 1. С. 94-97. URL: https://elibrary.ru/item.asp?id=27053023.

3. Золотин Андрей Алексеевич, Левенец Даниил Григорьевич, Зотов Михаил Анатольевич, Бирилло Анастасия Игоревна, Березин Алексей Иванович, Иванова Анна Валерьевна, Тулупьев Александр Львович, Алгоритмы обработки и визуализации алгебраических байесовских сетей // Образовательные технологии и общество. 2017. №1 С.446-457.

4. Зотов М.А., Иванова А.В. Алгебраические байесовские сети: статистический анализ сложности алгоритмов синтеза минимального графа смежности и его корректность // СПИСОК-2017: Материалы всероссийской научной конференции по проблемам информатики. Санкт-Петербург, 2017.

5. Зотов М.А., Левенец Д.Г., Иванова А.В., Тулупьев А.Л. Статистическая сложность синтеза вторичной структуры алгебраических байесовских сетей: двухфакторный анализ // ГИБРИДНЫЕ И СИНЕРГЕТИЧЕСКИЕ ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ. (Светлогорск, 6-11 июня 2016 г.). Светлогорск: Балтийского федерального университета им. Иммануила Канта, 2016. С. 405-411. URL: http://elibrary.ru/item.asp?id=26130319.

6. Berezin A.I., Ivanova A.V., Zotov M.A. Minimal Join Graphs' Set Synthesis: Performance Statistical Estimate of the Decremental Algorithm // International Conference on Soft Computing and Measurements (SCM). (St. Petersburg, 25-27 May 2016). St. Petersburg: IEEE, 2016.

C. 42-44. URL: http://ieeexplore.ieee.org/document/7519677/.

Список литературы

[1] Березин А.И., Зотов М.А., Иванова А.В. Синтез множества минимальных графов смежности: статистическая оценка сложности инкрементального алгоритма // СПИСОК-2016: Материалы всероссийской научной конференции по проблемам информатики. Санкт-Петербург, 2016. С. 443–453.

[2] Березин А.И., Иванова А.В., Зотов М.А. Синтез множества минимальных графов смежности: статистическая оценка сложности декрементального алгоритма // Международная конференция по мягким вычислениям и измерениям. 2016. Вып. Секции 1-3. Т. 1. С. 94-97. URL: https://elibrary.ru/item.asp?id=27053023.

[3] Березин А.И., Тулупьев А.Л. Система инкрементального и декрементального синтеза множества минимальных графов смежности алгебраических байесовских сетей Algebraic Bayesian Network Minimal Join Graph Set Incremental and Decremental Constructor, Version 01 for CSharp (AlgBN MJGS I&DC cs.v.01). Свид. о госуд. регистрации No 2016617395 (04.07.2016). Роспатент.

[4] 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.

[5] Золотин Андрей Алексеевич, Левенец Даниил Григорьевич, Зотов Михаил Анатольевич, Бирилло Анастасия Игоревна, Березин Алексей Иванович, Иванова Анна Валерьевна, Тулупьев Александр Львович, Алгоритмы обработки и визуализации алгебраических байесовских сетей // Образовательные технологии и общество. 2017. №1 С.446-457.

[6] Зотов М.А., Иванова А.В. Алгебраические байесовские сети: статистический анализ сложности алгоритмов синтеза минимального

графа смежности и его корректность // СПИСОК-2017: Материалы всероссийской научной конференции по проблемам информатики. Санкт-Петербург, 2017.

[7] Зотов М.А., Левенец Д.Г., Иванова А.В., Тулупьев А.Л. Статистическая сложность синтеза вторичной структуры алгебраических байесовских сетей: двухфакторный анализ // ГИБРИДНЫЕ И СИНЕРГЕТИЧЕСКИЕ ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ. (Светлогорск, 6-11 июня 2016 г.). Светлогорск: Балтийского федерального университета им. Иммануила Канта, 2016. С. 405-411. URL: http://elibrary.ru/item.asp?id=26130319

[8] 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. Т. 16. № 1. С. 122–132.

[9] Зотов М.А., Тулупьев А.Л. Вторичная структура алгебраических байесовских сетей: статистическая оценка сложности прямого алгоритма синтеза // SCM'2015: XVIII Международная конференция по мягким вычислениям и измерениям. Санкт-Петербург, 19-21 мая 2015 г. С. 158-162.

[10] Зотов М.А., Тулупьев А.Л. Синтез вторичной структуры алгебраических байесовских сетей. // Компьютерные инструменты в образовании, 2015. 1. С. 3–16.

[11] Зотов М.А., Тулупьев А.Л. Синтез вторичной структуры алгебраических байесовских сетей: методика статистической оценки сложности и компаративный анализ прямого и жадного алгоритмов // Компьютерные инструменты в образовании, 2015. № 1. С. 3–16.

[12] Зотов М.А., Тулупьев А.Л. Синтез вторичной структуры алгебраических байесовских сетей: сравнительный анализ статистических

оценок сложности двух алгоритмов // VIII международная научно-практическая конференция «Интегрированные модели и мягкие вычисления в искусственном интеллекте». Т. 2. С. 790-798.

[13] 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.

[14] Критерий Шапиро-Уилка [Electronic resource] // machinelearning.ru. URL: http://www.machinelearning.ru/wiki/index.php?title = Критерий_Шапиро-Уилка (accessed at 30.04.2017)

[15] Левенец Д.Г., Зотов М.А. Декрементальный синтез вторичной структуры алгебраических байесовских сетей: статистическая оценка сложности // СПИСОК-2016: Материалы всероссийской научной конференции по проблемам информатики. Санкт-Петербург, 2016.

[16] Левенец Д.Г., Зотов М.А., Тулупьев А.Л. Инкрементальный алгоритм синтеза минимального графа смежности // Компьютерные инструменты в образовании. 2015. Вып. 6. С. 3–18. URL: http://ipo.spb.ru/journal/index.php?article/1790/.

[17] Левенец Д.Г., Тулупьев А.Л. Система инкрементального и декрементального синтеза вторичной структуры алгебраических байесовских сетей Algebraic Bayesian Networks 2ry Structure Incremental and Decremental Constructor, Version 01 for CSharp (AlgBN 2 SID Constructor cs.v.01) (Свидетельство). Свид. о госуд. регистрации No 2016618756 (05.08.2016). Роспатент.

[18] Мальчевская Е.А., Тулупьев А.Л.Система представления фрагментов знаний алгебраических байесовских сетей Algebraic Bayesian Networks Knowledge Patterns Modeler, Version 01 for CSharp (AlgBN

KP Modeler cs.v.01) (Свидетельство). Свид. о госуд. регистрации No 2016617214 (29.06.2016). Роспатент.

[19] Новиков Ф.А. Дискретная математика: учебник для вузов. 2-е изд. Стандарт третьего поколения. СПб.: Питер, 2013. С. 432.

[20] 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.

[21] Сироткин А.В. Локальный априорный вывод в алгебраических байесовских сетях: комплекс основных алгоритмов // Труды СПИИРАН. Вып.5. СПб.: Наука, 2007. С. 100-111.

[22] 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.

[23] Сироткин А.В. Проверка и поддержание непротиворечивости алгебраических байесовских сетей: вычислительная сложность алгоритмов // Труды СПИИРАН. 2010. Вып.4(15). С.162-192.

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

[25] Тулупьев А.Л. Алгебраические байесовские сети: логико-вероятностная графическая модель баз фрагментов знаний с неопределенностью. Санкт-Петербургский институт информатики и автоматизации РАН", — Изд-во С.-Петерб. ун-та, 2009.

[26] Тулупьев А.Л. Алгебраические байесовские сети: логико-вероятностный подход к моделированию баз знаний с неопределенностью. // СПб.: СПИИРАН, 2000. 282 с.

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

[28] Тулупьев А.Л. Ациклические алгебраические байесовские сети: логико-вероятностный вывод // Нечеткие системы и мягкие вычисления. 2006. Т. 1. № 1. С. 57–93.

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

[30] 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.

[31] Тулупьев А.Л. Задача локального автоматического обучения в алгебраических байесовских сетях: логико-вероятностный подход // Труды СПИИРАН. 2008. № 7. С. 11–25.

[32] Тулупьев А.Л. Непротиворечивость оценок вероятностей в алгебраических байесовских сетях // Вестник СПбГУ. 2009. № 3. С. 144–151.

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

[34] Тулупьев А.Л., Сироткин А.В. Алгебраические байесовские сети: принцип декомпозиции и логико-вероятностный вывод в условиях неопределенности // Информационно измерительные и управляющие системы. 2008. Т. 6. № 10. С. 85–87.№ 1. С. 139–142.

[35] Тулупьев А.Л., Сироткин А.В. Программа для моделирования фрагмента знаний алгебраической байесовской сети, поддержания его непротиворечивости и апостериорного вывода в нем Algebraic Bayesian Network Knowledge Pattern Modeler, Reconciler

and Propagator Version 01 for C++ (AlgBN KP MRP cpp.v.01) (Свидетельство). Свид. о гос. регистрации программы для ЭВМ. Рег. № 2010615242(13.08.2010).Роспатент.

[36] Тулупьев А.Л. Система представления алгебраических байесовских сетей и их фрагментов Algebraic Bayesian Networks Modeler, Version 01 for Java (AlgBN Modeler j.v.01) (Свидетельство). Свид. о гос. рег. прогр. для ЭВМ. Рег. № 2009613802 (16.07.2009). Роспатент.. Бюлл. «Прогр. для ЭВМ, БД, топол. инт. микросх.». 2009. № 4. С. 64–65.

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

[38] 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.))

[39] Фильченков А.А., Тулупьев А.Л. Связность и ацикличность первичной структуры алгебраической байесовской сети // Вестник Санкт-Петербургского университета. Серия 1: Математика. Механика. Астрономия. 2013. № 1. С. 110–119.

[40] Фильченков А.А., Тулупьев А.Л. Третичная структура алгебраической байесовской сети // Труды СПИИРАН. 2011. № 18. С. 164–187.

[41] Фильченков А.А., Тулупьев А.Л., Сироткин А.В. Минимальные графы смежности алгебраической байесовской сети: формализация основ синтеза и автоматического обучения // Нечеткие системы и мягкие вычисления. 2011. Т. 6. № 2. С. 145–163.

[42] Фильченков А.А., Тулупьев А.Л., Сироткин А.В. Особенности анализа вторичной структуры алгебраической байесовской сети // Труды СПИИРАН. 2010. № 1 (12). С. 97–118.

[43] Angular 2 [Electronic resource] // One framework. - Angular. URL: https://angular.io/ (accessed at 23.04.2017).

[44] BitBucket [Electronic resource]. URL: https://bitbucket.org/ (accessed at 30.04.2017).

[45] D3 [Electronic resource] // D3.js labrary | D3. URL: https://d3js.org/ (accessed at 29.04.2017).

[46] Dart [Electronic resource] // Dart programming language | Dart. URL: https://www.dartlang.org/ (accessed at 29.04.2017).

[47] Levenets D.G., Zotov M.A., Romanov A.V., Tulupyev A.L., Zolotin A.A., Filchenkov A.A. Decremental and Incremental Reshaping of Algebraic Bayesian Networks Global Structures // Proceedings of the First International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'16). T. 2. Sochi: Springer, 2016. C. 57-67. URL: http://link.springer.com/book/10.1007/978-3-319-33816-3.

[48] RStudio – Open source and enterprise-ready professional software for R [Electronic resource] // r-project.org. URL: https://www.rstudio.com/ (accessed at 27.04.2017).

[49] The R Project for Statistical Computing [Electronic resource] // r-project.org. URL: https://www.r-project.org/ (accessed at 23.04.2017).

[50] Jakobsen Thomas. Advanced character physics // IN PROCEEDINGS OF THE GAME DEVELOPERS CONFERENCE 2001. 2001. P. 19.

[51] TypeScript [Electronic resource] // TypeScript - JavaScript that scales. URL: https://www.typescriptlang.org/ (accessed at 23.04.2017).