Компьютерные инструменты в образовании, 2015
№ 1: 3–18
УДК: 004.8
http://ipo.spb.ru/journal

информатика

СИНТЕЗ ВТОРИЧНОЙ СТРУКТУРЫ АЛГЕБРАИЧЕСКИХ БАЙЕСОВСКИХ СЕТЕЙ: МЕТОДИКА СТАТИСТИЧЕСКОЙ ОЦЕНКИ СЛОЖНОСТИ И КОМПАРАТИВНЫЙ АНАЛИЗ ПРЯМОГО И ЖАДНОГО АЛГОРИТМОВ*

Зотов Михаил Анатольевич, Тулупьев Александр Львович

Аннотация

В статье рассматриваются алгоритмы прямого и жадного синтеза минимального графа смежности. Проведен сравнительный статистический анализ времени работы указанных алгоритмов на основе вычислительных экспериментов со специально сгенерированными наборами входных данных. Для генерации тестовых данных был разработан алгоритм генерации нагрузок вершин графа смежности с заданными характеристиками. 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, а в под диапазоне 35–60 выигрыш в скорости зависит от конкретного набора данных. Кроме того, можно ожидать, что в диапазоне 5–60 будет обнаружено некоторое число статистических выбросов, сигнализирующих об особенностях в соответствующих наборах исходных данных. Статья также имеет дидактическую цель: обеспечить читателя последовательным описанием мотивированного применения ряда статистик в задаче сравнения двух алгоритмов и особенностей интерпретации полученных показателей. Статистический подход к сравнительной оценке мотивирован тем, что теоретические оценки сложности обсуждаемых алгоритмов не очевидны, зависят от скрытых особенностей исходных данных и не вычисляются по ним непосредственно.

Ключевые слова: *представление неопределенности, алгебраические байесовские сети, вероятностные графические модели, фрагмент знаний, знания с неопределенностью, логико-вероятностный вывод, статистическое исследование сложностей алгоритмов.*

* Статья содержит материалы исследований, частично поддержанных грантом РФФИ 15-01-09001 — «Комбинированный логико-вероятностный графический подход к представлению и обработке систем знаний с неопределенностью: алгебраические байесовские сети и родственные модели».

1. ВВЕДЕНИЕ

В области исследования вероятностных графических моделей (ВГМ) одним из актуальных направлений является развитие автоматического обучения (машинного обучения) таких моделей, а также алгоритмов для него [1, 4]. Это обусловлено, с одной стороны, тем, что байесовские сети доверия (и родственные модели) оказались весьма востребованными в ряде областей [8, 18, 20], однако усилия по формированию сети в каждом индивидуальном случае оказываются весьма значительными. С другой стороны, за счет автоматизации широчайшего спектра отраслей человеческой деятельности с помощью компьютерных технологий стали накапливаться колоссальные массивы данных, которые можно было бы использовать при автоматическом обучении разнообразных графических моделей. В силу особенностей структуры таких моделей, алгоритмы их автоматического обучения по своему объекту синтеза делятся на два класса: локальные — когда требуется синтезировать параметры модели (условные вероятности, совместные вероятности, оценки вероятности небольшого числа тесно связанных событий или утверждений), глобальные — когда требуется синтезировать структуру модели. Оба класса алгоритмов для каждого класса ВГМ обладают своими особенностями. Были достигнуты определенные успехи в развитии теории и алгоритмов автоматического обучения [5], однако в указанном направлении исследований остается много нерешенных задач. Возможно, круг таких задач окажется неисчерпаем.

Алгебраические байесовские сети (АБС) являются одним из классов ВГМ [2, 6, 7]. В настоящее время в качестве основной глобальной структуры АБС выступают графы смежности [9, 13, 14]. При подготовке к решению проблем, связанных с автоматическим обучением АБС, оказалось необходимым уделить особое внимание задаче синтеза вторичной структуры АБС по заданному набору вершин — фактически речь идет об алгоритмизации одного из этапов глобального обучения АБС [1].

Поскольку было доказано, что пространство графов смежности может быть описано матроидом [10], одним из очевидных решений построения такого графа является применение жадного алгоритма [1]. Вместе с тем, был предложен другой алгоритм — прямой алгоритм [9], существенно отличающийся от жадного. В отношении обоих алгоритмов построены теоретические оценки сложности, но они оказались неприменимы на практике, поскольку опираются на латентные свойства объектов: численные характеристики, которые требуются для оценки сложности работы на конкретном наборе данных (наборе вершин — максимальных конъюнктов АБС), становятся доступными только после построения искомого объекта — графа смежности, причем еще требуются дополнительные усилия по его анализу [11, 15]. В такой ситуации представляется перспективным один из вариантов эмпирического подхода — проведение вычислительных экспериментов и построение на основе обработки их результатов абсолютных и относительных статистических оценок времени исполнения конкурирующих алгоритмов. Построение таких статистических оценок по сгенерированным особым образом наборам исходных данных составляет научную *цель настоящей работы*, которая сочетается с дидактической целью — описать методику построения и компаративного анализа статистической оценки сложности двух алгоритмов, проиллюстрировав ее конкретными примерами.

Указанные цели требуют некоторых дополнительных пояснений. Следует отметить, что зачастую компаративный статистический анализ времени исполнения конкурирующих алгоритмов производится достаточно поверхностно [16, 17], сводясь, в лучшем случае, к вычислению отношения средних скоростей работы. Такая точечная оценка

ключевого показателя, хотя и является наиболее интересной, все же недостаточно характеризует положение дел: в частности, преимущества сравниваемых алгоритмов могут разниться на различных наборах данных или же не быть статистически значимыми.

Чтобы избежать этих и некоторых других недостатков в настоящей работе, для сравнения временных сложностей двух алгоритмов построения минимальных графов смежности предпринимаются несколько шагов. Во-первых, замеряются средние времена выполнения генерации минимальных графов смежностей над одной и той же совокупностью исходных данных. Это позволяет минимизировать влияние случайных внешних факторов на оценку времени выполнения. Во-вторых, рассматривается не только соотношение получившихся средних времен выполнения алгоритмов (среднее и медиана), но также строятся особые статистики (дисперсия, первый и третий квартиль, первый и девятый дециль), позволяющие оценить и визуализировать характеристики разброса. Совместное использование в анализе среднего, медианы и других статистик позволяет в указанном контексте отсечь случайные особенности входных данных, представить более ясную картину поведения алгоритмов, а также выявить смещения и выбросы.

Кроме того, выявленные выбросы представляют особый интерес для дальнейшего исследования пространства входных данных — они позволяют отобрать такие совокупности исходных данных, на которых алгоритмы ведут себя особо «хорошо» и особо «плохо». Именно на основе сведений о таких совокупностях можно проводить дальнейшую оптимизацию алгоритмов, а также, возможно, ставить новые задачи в развитии теории графов смежности, нацеленные на поиск признаков в исходных данных, отвечающих за особо высокую или особо низкую скорость генерации искомого графа. Пространство графов смежности имеет очень высокую мощность [15] и обладает рядом неизученных особенностей, поэтому на текущем этапе развития теории алгебраических байесовских сетей представляется целесообразной случайная генерация совокупностей исходных данных для последующего построения графов. При этом нужно обеспечить ненулевую вероятность генерации каждой возможной совокупности данных. Отметим, что если бы мы располагали сведениями из некоторой конкретной предметной области, то при случайной генерации таких совокупностей данных выдвигалось бы иное требование — их репрезентативная генерация, иначе говоря и немного упрощая, потребовалось бы обеспечить, чтобы совокупность данных, которая чаще встречается в предметной области, имела большую вероятность сгенерироваться, чем совокупность, которая в предметной области встречается реже.

2. ОПРЕДЕЛЕНИЯ И ОБОЗНАЧЕНИЯ

Воспользуемся системой терминов и обозначений, сложившихся в [2, 6–9].

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 ребер. Пусть имеется набор вершин $V = \{v_1, v_2, \ldots, v_n\}$ и такие нагрузки $W = \{W_1, W_2, \ldots, W_n\}$, причем $\forall u \in V \quad W_u$ является нагрузкой для вершины u.

Определение 1. Назовем неориентированный граф $G = \langle V, E \rangle$ 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 $W_u \cap W_v \subseteq W_s$.
2. $\forall \langle u, v \rangle \in E \quad W_u \cap W_v \neq \emptyset$.

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

Определение 2. Пересечение нагрузок двух вершин $W_u \cap W_v$ будем называть *сепаратором*.

Определение 3. Число элементов в заданном сепараторе назовем *мощностью сепаратора*.

Определение 4. Будем называть *списком* набор объектов, доступных по индексу.

Определение 5. Пусть дан граф $G = \langle V, E \rangle$, тогда граф $G \downarrow q = \langle V', E' \rangle : V' = \{v : v \in V \ \& \ q \in W_v\}, E' = \{\langle p, q \rangle : p, q \in V' \ \& \ \langle p, q \rangle \in E\}$.

Граф смежности, определенный таким образом, обеспечивает свойство *магистральной связности* [15]. Оно важно в теории алгебраических байесовских сетей [13], где графы смежности нашли свое применение в качестве вторичной глобальной структуры таких сетей. Вкратце, магистральная связность графа смежности обеспечивает корректность распространений изменений, произошедших в одном узле алгебраической байесовский сети, на все ее остальные узлы, причем такое распространение не выполняется «глобально», что было бы вычислительно «дорого», а сводится к передаче изменений между соседними узлами, то есть позволяет обходиться на каждом шаге «дешевыми» локальными вычислениями [3]. Напомним, что графы смежности применяются также в теории реляционных баз данных [19]: граф связей таблиц в реляционной базе данных — это один из видов графов смежности.

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

Q — упорядоченная по убыванию мощностей сепараторов очередь [9] уникальных элементов, рассматриваемая как некоторое множество. Генерация множества Q осуществляется следующим образом: для любых двух вершин из V вычисляется их сепаратор, затем из полученного набора сепаратором в множество Q добавляются сепараторы в порядке убывания их мощностей. Может возникнуть ситуация, когда в наборе сепараторов будут содержаться одинаковые сепараторы, в таком случае в множество Q необходимо добавить только один из них (см. пример 1). Очередь реализуется в языках программирования C#, C++ и Java с помощью Queue [21–24].

S — структура данных, реализующая представление множества. Элементами S являются вершины. Порядок элементов в S не важен, значит, для реализации представления множества можно использовать структуру данных список. Для реализации такого списка в языках программирования C# и Java можно использовать List [25–27], в C++ — vector [28].

Delegate(S) — функция, возвращающая произвольный элемент множества S. При этом сам возвращаемый элемент не удаляется из множества S. Если S пусто, алгоритм предусматривает добавление новых элементов в множество. Таким образом, функция Delegate всегда будет принимать на вход непустое множество.

Component(G, v, q) — функция, которая принимает в качестве первого аргумента граф, второго — вершину, третьего — сепаратор (нагрузку). Первым шагом своей работы функция генерирует граф $G \downarrow q$, затем получает множество достижимых вершин в графе $G \downarrow q$ из вершины v, включая саму эту вершину. Таким образом, функция вернет множество достижимых вершин, содержащее как минимум вершину v.

Q.Top() — функция, которая возвращает очередной элемент из очереди Q и удаляет этот элемент из Q.

newNode(weight) — функция, которая создает новую вершину с нагрузкой weight.

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

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

edge.RemoveAllowed — метка, которая показывает, возможно ли удаление ребра edge или нет. Начальное значение — true.

getLength(A) — функция, которая возвращает число элементов структуры A.

asInt(number) — функция, которая возвращает округленное до ближайшего целого число number.

Пример. Пусть имеется $V = \{A, B, C, D, E, F\}$ — набор вершин, причем $W_A = \{a, b, c\}$, $W_B = \{a, e\}, W_C = \{a, b, d\}, W_D = \{d, f\}, W_E = \{a, q\}, W_F = \{a, w\}$. Тогда множество Q будет иметь вид $\{\{a, b\}, \{d\}, \{a\}\}$. Заметим, что сепаратор $\{a\}$ формируется на пересечении нагрузок нескольких вершин, например: $W_A \cap W_B = W_A \cap W_E = \{a\}$, но в множество Q сепаратор $\{a\}$ вошел только один раз.

3. АЛГОРИТМЫ ПОСТРОЕНИЯ ГРАФА СМЕЖНОСТИ

3.1. Прямой алгоритм

В листинге 1 приведен алгоритм прямой генерации минимального графа смежности, который был формализован в статье [9]. Алгоритм состоит из двух основных частей.

Algorithm 1 Прямой алгоритм построения минимального графа смежности [9]

input: V, W
output: G = <V,E>

```
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 W v && (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
```

В строках 2–7 формируется множество сепараторов. В строках 8–18 происходит формирование минимального графа смежности. Рассмотрим вторую часть алгоритма подробнее.

В строке 9 извлекается очередной сепаратор q, далее, для каждой вершины (строка 11) проверяется вхождение сепаратора q в нагрузку текущей вершины v. Если нагрузка вершины v содержит сепаратор q, то вычисляется множество достижимых вершин (при пустом S). Такое множество будет содержать элементы, между которыми есть путь, причем в нагрузку каждой вершины будет входить сепаратор q. Вычислив множество S, на одном из следующих шагов алгоритм соединит с одной из этих вершин текущую (строки 14, 16), таким образом будет формироваться связность между вершинами, в нагрузки которых входит сепаратор q.

Пройдя по всем сепараторам q из множества Q, алгоритм гарантированно соединит вершины, сепараторы которых не пусты. Минимальность полученного графа смежности обеспечивается тем, что в строке 12 происходит проверка того, что текущая вершина, содержащая сепаратор q, не входит в множество достижимых вершин, которые также содержат сепаратор q. Таким образом, такая вершина не будет соединена ни с одной из вершин такого множества, а значит, не будут проведены никакие дополнительные, лишние ребра.

Следует особо оговорить, что изменение порядка следования сепараторов одинаковой мощности в Q может изменить число шагов алгоритма и набор ребер в результирующем минимальном графе смежности. В настоящей работе не изучается вопрос генерации такой структуры множества Q, при которой достигается минимум числа шагов алгоритма, не изучается также вопрос генерации всех минимальных графов смежности.

3.2. Жадный алгоритм

В листинге 2 приведен жадный алгоритм генерации минимального графа смежности. В отличие от прямого алгоритма генерации минимального графа смежности, который основывается на постепенном добавлении необходимых ребер, жадный алгоритм вначале добавляет все возможные ребра (строки 3–6), после чего жадным образом избавляется от ребер, удаление которых не нарушают магистральную связность. В частности, в строках 9–12 мы выбираем еще не проверенное ребро (метка которого имеет значение true), если непроверенных ребер не осталось, то построение графа закончено (строки 13–14). В противном случае в строке 15 проверяется возможность удаления данного ребра без потери магистральной связности. Если функция PathExists возвращает значение true, то данное ребро удаляется, в противном случае ребро из графа не удаляется, а значение метки ребра выставляется в false. Так как множество графов смежности формирует матроид [10], а удаление допустимого ребра — это переход от одного графа к другому с уменьшением числа ребер, то по свойствам матроида алгоритм сойдется к минимальному графу смежности.

4. АЛГОРИТМ ГЕНЕРАЦИИ ТЕСТОВЫХ НАБОРОВ ДАННЫХ

Для проведения тестирования алгоритмов нам необходимо сформировать набор тестовых данных и подать его на вход прямому и жадному алгоритмам. Для этого был разработан алгоритм формирования наборов вершин с нагрузками, учитывающий необхо-

Algorithm 2 Жадный алгоритм построения минимального графа смежности

input: V, W **output**: G = <V,E>

```
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
```

димую количественную и структурную специфику, которой должны обладать выходные данные (листинг 3). На выход алгоритму генерации тестовых наборов данных подаются:

- число генерируемых вершин;
- алфавит, по которому будет осуществляться генерация множества нагрузок;
- диапазоны мощностей нагрузок и их процентное содержание среди общего числа вершин.

Разработанный алгоритм позволяет в определенной степени контролировать получающиеся тестовые наборы данных с учетом трех указанных показателей.

Для генерации одного тестового набора данных используется нижеприведенный подход. Следующие шаги алгоритм выполняет для каждого диапазона мощностей нагрузок (строка 3), поданного на вход алгоритму. В строке 5 алгоритм рассчитывает число countNodes, равное числу вершин для данного диапазона мощностей нагрузок. В строках 6–14 осуществляется генерация вершин: высчитывается число count (строка 8) — оно определяет количество элементов нагрузки данной вершины; далее, в строках 9–13 генерируются элементы нагрузки данной вершины. Из алфавита, переданного в качестве параметра алгоритму, случайным образом извлекается нагрузка и приписывается текущей вершине (строки 11, 13); генерация осуществляется до тех пор, пока число элементов нагрузки данной вершины не станет равным count. После генерации нагрузки данной вершины эта вершина добавляется в множество вершин данного диапазона мощностей нагрузок. Когда число вершин для данного диапазона мощностей нагрузок

Algorithm 3 Алгоритм генерации тестовых наборов данных

input: N, alphabet, distributions
output: $V = \{v_1, v_2, \ldots, v_N\}$

```
 1: function TestSets
 2:     nodes = ∅
 3:     foreach (distrib in distributions)
 4:         subNodes = ∅
 5:         countNodes = asInt(N * distrib.percent)

 6:         while (getLength(subNodes) ≠ countNodes)
 7:             weightOfNode = ∅
 8:             count = getRandom(distrib.min, distrib.max)

 9:             while (getLength(weightOfNode) ≠ count)
10:                 index = getRandom(0, getLength(alphabet))
11:                 elementWeight = alphabet[index]

12:                 if (elementWeight not in weightOfNode) then
13:                     weightOfNode = weightOfNode ∪ elementWeight

14:             subNodes = subNodes ∪ newNode(weightOfNode)

15:         nodes = nodes ∪ subNodes

16:     return nodes
```

станет равным countNodes, алгоритм добавит полученное множество вершин к общему множеству вершин (строка 15) и продолжит генерацию для следующего диапазона мощностей нагрузок.

Предполагается, что у каждого элемента алфавита существует ненулевая вероятность попасть в нагрузку вершины, следовательно, так как выборка элементов алфавита осуществляется независимо, делается вывод, что возможно любое сочетание нагрузок вершин, что и нужно для исследования относительных сложностей алгоритмов. Таким образом, разработанный алгоритм позволяет контролировать получающиеся тестовые наборы данных с учетом показателей, передаваемых в качестве параметров.

5. ОПИСАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ

Прямой и жадный алгоритмы были реализованы на языке программирования C# для статистического исследования их временной сложности. Кроме набора данных, передаваемых на вход алгоритму синтеза минимального графа смежности, влияние на время его работы оказывают некоторые случайные факторы, такие, например, как состояние операционной системы, в которой запущено приложение, реализующее данный алгоритм. Чтобы корректно сравнить реализации таких алгоритмов, требуется обеспечить решение следующих трех задач. Задача первая — предложить способ снижения влияния случайных факторов; вторая — предложить способ, который позволит обрабатывать вы-

борки, у которых имеются сходные свойства. Задача третья — компактно представить результаты обработки статистических сложностей двух алгоритмов. Решение первых двух задач позволит давать более точную оценку сложностям алгоритмов по сравнению с оценкой, получаемой на основе единичных испытаний. Решение третей задачи — наглядное представление результатов вычислений: представление, удобное для восприятия, трактовки и дальнейшего исследования.

5.1. Усредненная оценка времени выполнения алгоритма над одним и тем же набором данных

Чтобы уменьшить влияние случайных факторов, связанных с деятельностью операционной системы, на оценку времени реализации алгоритма каждый набор данных передается алгоритму синтеза минимального графа смежности m раз (в настоящей работе $m = 50$), затем рассчитывается среднее время обработки набора данных. Таким образом, усреднение времени выполнения алгоритма синтеза минимального графа смежности над каждым набором данных позволяет снизить влияние случайных факторов.

5.2. Усредненная оценка времени выполнения алгоритма над совокупностью наборов данных

Кроме случайных факторов, на время выполнения синтеза минимального графа влияет сам набор данных. С помощью алгоритма генерации тестовых наборов данных мы формируем ряд наборов (всего $n = 60$ наборов) с такими же свойствами (одинаковое число вершин, одинаковое распределение мощностей нагрузок). На основе полученных оценок времени по двум алгоритмам на каждом наборе вычисляются отношение средних времен и натуральный логарифм отношения средних времен. Таким образом, у нас формируется выборка в 60 элементов из отношений средних времен и выборка в 60 элементов из натуральных логарифмов таких отношений. Для первой выборки мы вычисляем среднее, медиану, 1 и 3 квартиль, 1 и 9 дециль. Для второй выборки мы вычисляем среднее, дисперсию, верхнюю и нижнюю границы доверительного интервала.

Среднее позволяет понять, как два алгоритма ведут себя на выборках в среднем; от медианы отличается тем, что при вычислении среднего учитываются все выбросы — особо долгие и особо быстрые вычисления, которые могут сильно повлиять на значение среднего. В наших экспериментах среднее и медиана практически неотличимы на графиках, поэтому мы отображаем только медиану; 1 и 3 квартиль позволяют обнаружить выбросы — особо «хорошие» выборки, а 1 и 9 дециль — особо «плохие» выборки. Дисперсия позволяет отследить отклонение отношений скоростей выполнения алгоритмов от среднего; верхняя и нижняя границы 95% доверительного интервала позволяют понять, в каких пределах должно лежать такое отношение скоростей. Также стоит обратить внимание на размахи — части графиков, лежащие между 1 и 3 квартилем и между 1 и 9 децилем. Анализ таких размахов позволяет делать выводы о выборках: чем меньше данные размахи, тем менее вероятно, что среди выборок найдутся такие, на которых алгоритмы будут работать особо «хорошо» или особо «плохо».

Таким образом, подход, описанный выше, позволяет представить соотношение времени работы двух алгоритмов на случайной выборке из 60 наборов исходных данных, в которой фиксируются число вершин и распределение мощностей нагрузок. Представить соотношения удается за счёт статистик (медиана, дисперсия и др.), позволяющих увидеть и выявить особенности работы алгоритмов.

5.3. Визуализация статистик на диапазоне вершин

Выполняя вычислительные эксперимнты для настоящей работы, мы ограничились рассмотрением в пространстве графов смежности интересующего нас диапазона числа вершин (от 5 до 105). С помощью алгоритма генерации тестового набора данных были созданы выборки для 5, 10, ..., 100, 105 вершин (шаг равен 5). В отношении каждого сгенерированного набора данных применялась процедура измерения статистической сложности, описанная в пункте 4.2.

5.4. Обозначения статистик и особенности их интерпретации

После замера статистической сложности двух конкурирующих алгоритмов отношения скоростей логарифмируются $\rho_i = \ln r_i$, затем рассчитываются среднее $\bar{\rho}_i$ и дисперсия s_i^2 величин ρ_i. В предположении, что логарифмы отношений распределены нормально, вычисляются нижняя и верхняя границы 95% доверительного интервала $\rho_i^{\pm} = \rho_i \pm 2s_i$. Наконец, рассчитываются экспоненты этих величин с получением новых: среднего геометрического отношений $r_{g,i} = \exp \bar{\rho}_i$, а также нижней и верхней границ 95% доверительного интервала отношений $r_i^{\pm} = \exp \rho_i^{\pm}$.

Вычислением только среднего или медианы ограничиться нельзя, ибо вывод, основанный только на таких показателях, может быть в конечном итоге неправильно генерализован на всю совокупность выборок. С целью более точной характеризации относительного времени работыт вычисляют среднее ($\bar{R}_i$), медиану (M_i), первый дециль (d_1, D1), первый квартиль (Q_1, Q1), третий квартиль (Q_3, Q3), девятый дециль (d_9, D9). В скобках вторым идет обозначение величин на графиках.

Квартили и децили вычисляются потому, что отдельный интерес представляет поиск выбросов, на которых относительное время работы алгоритмов существенно отличается от среднего. Это говорит о том, что нам встретился набор данных, очень «удобный» для одного алгоритма и не очень «удобный» для второго. Выявление свойств таких наборов данных откроет возможность в дальнейшем выбирать оптимальный по прогнозируемому времени работы алгоритм. Кроме того, представляется интересным исследование пространства таких данных, ибо, возможно, при таком исследовании удастся выявить какие-либо априорные [2, 7, 8] свойства и оценки генерируемого минимального графа смежности. Однако из-за того, что усилия по исследованию априорных показателей остаются весьма значительными, задача выбора подхода к такому исследованию является актуальной. В настоящей статье не изучаются методы выявления и исследования априорных свойств тестовых данных с особенностями.

6. РЕЗУЛЬТАТЫ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ

Эксперименты показали, что различия между $r_{g,i}$, M_i, $\bar{R}_i$ незначительны, поэтому на графиках отображается только первая величина (на графике обозначена как Rg) наряду с d_1, Q_1, Q_3, d_9, а также с r_i^- и r_i^+.

На рис. 1–7 представлены графики указанных статистик относительного времени выполнения алгоритмов на разных наборах данных. На горизонтальной оси указано число вершин, по которым была получена данная статистика. На вертикальной оси откладывается величина статистики.

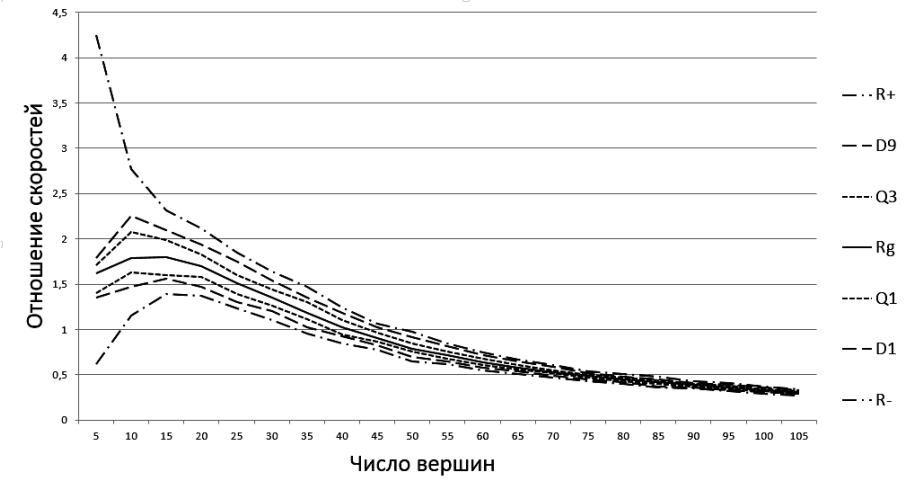

Рис. 1. Алфавит 52 символа, вершины 3–5-го порядков

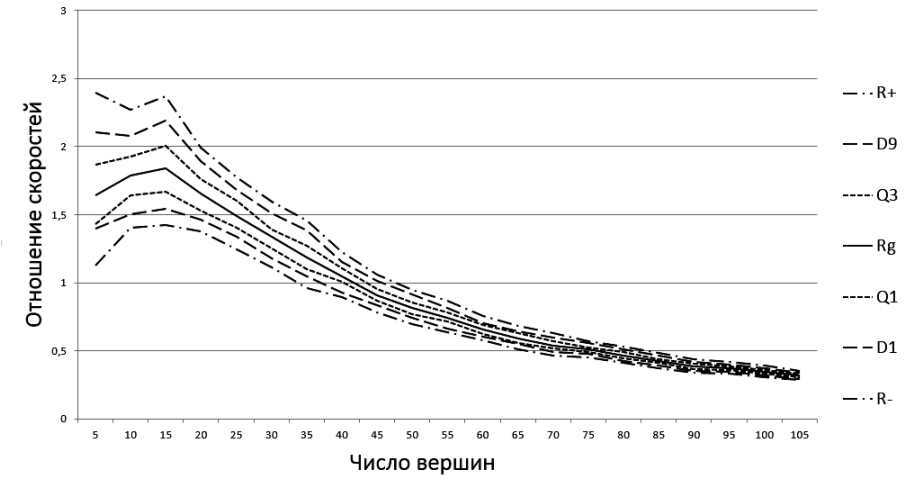

Рис. 2. Алфавит 52 символа, вершины 3–5-го порядков, 5% — 6–9 порядков

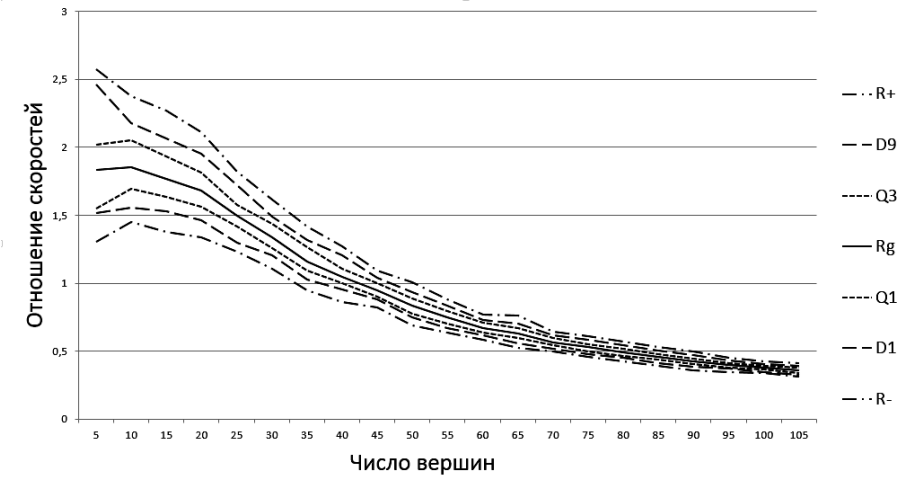

Рис. 3. Алфавит 52 символа, вершины 3–5-го порядков, 15% — 6–9 порядков

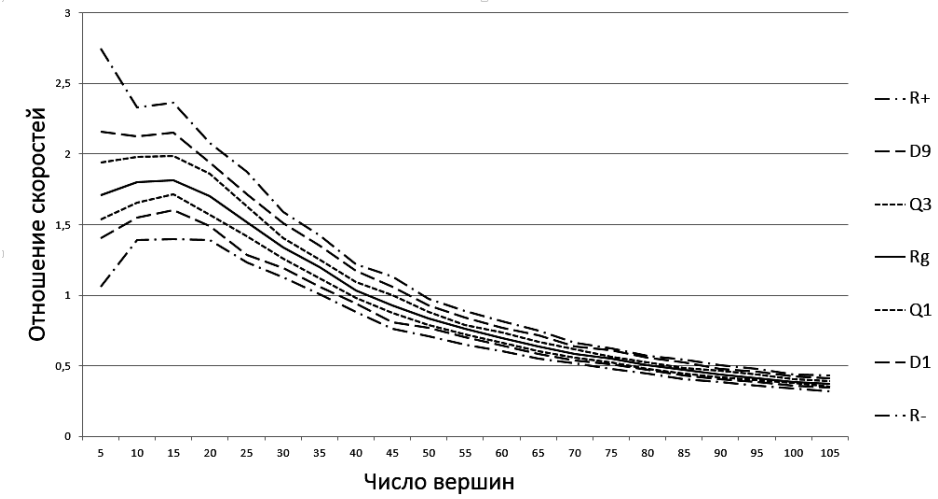

Рис. 4. Алфавит 52 символа, вершины 3–5-го порядков, 20% — 6–9 порядков

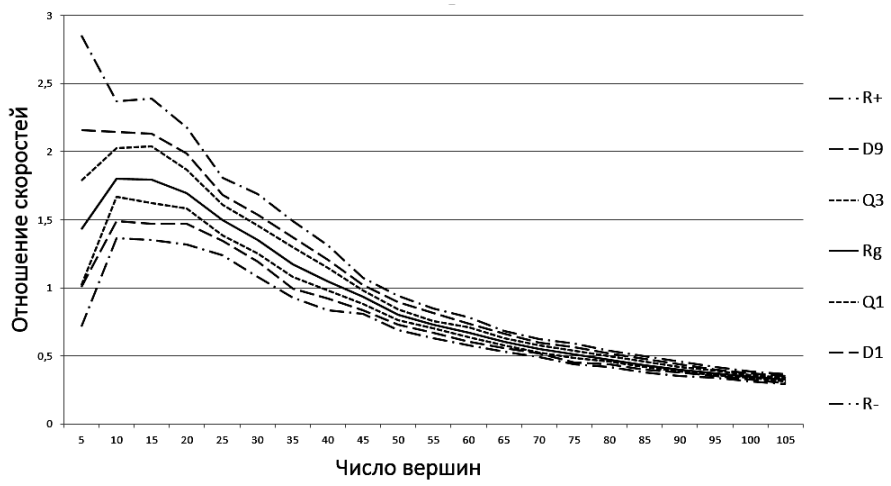

Рис. 5. Алфавит 52 символа, вершины 3–5-го порядков, 5% — 8–9 порядков

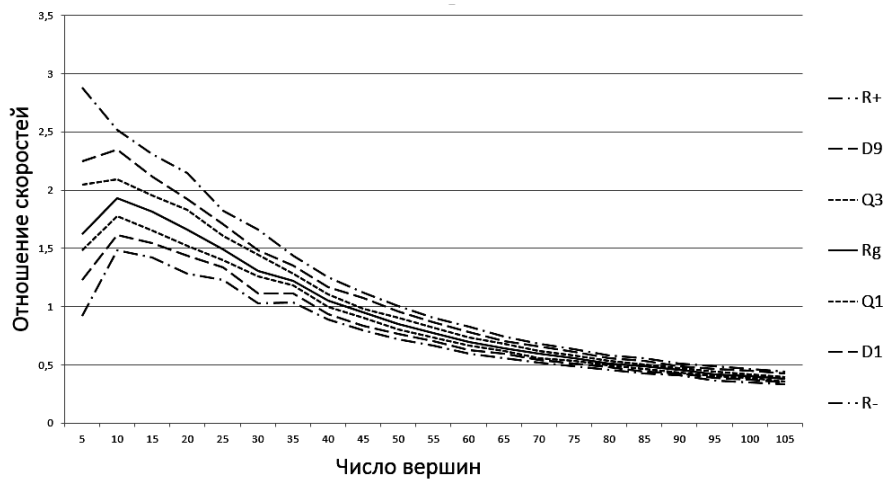

Рис. 6. Алфавит 52 символа, вершины 3–5-го порядков, 15% — 8–9 порядков

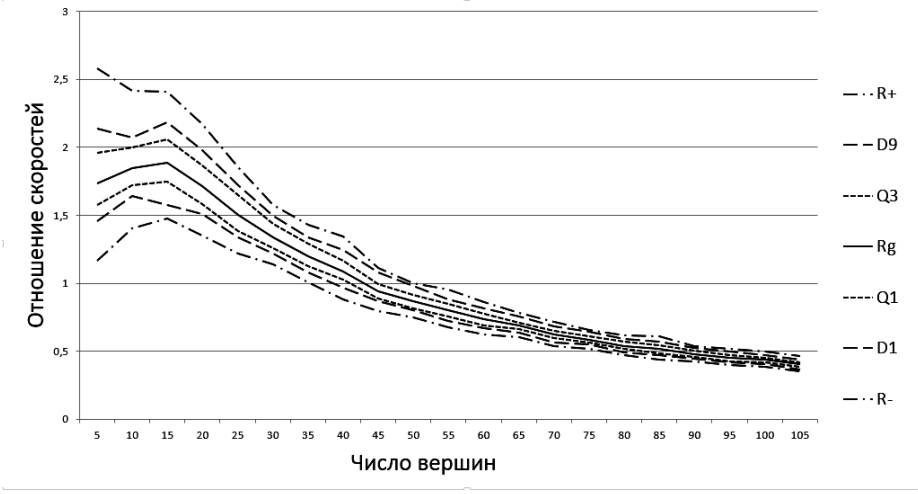

Рис. 7. Алфавит 52 символа, вершины 3–5-го порядков, 20% — 8–9 порядков

Для удобства интерпретации полученных результатов стоит вспомнить, что в диапазоне $Q_1 - Q_3$ лежит 50% значений, в $d_1 - d_9$ — 80%, а в $r^- - r^+$ — более 95% (при наших предположениях о нормальности распределения логарифмов отношений).

Легко увидеть, что выбранный для исследования диапазон числа вершин делится на три поддиапазона, в которых ситуации качественно различаются.

В поддиапазоне 5–35 жадный алгоритм работает существенно быстрее прямого, может быть, за исключением самого начала этого поддиапазона. В поддиапазоне 60–105, наоборот, прямой алгоритм работает существенно быстрее жадного. А в среднем поддиапазоне 35–60 имеет место переходная ситуация; видимо, в этом случае для выбора алгоритма потребуется провести дополнительные исследования, чтобы учесть более широкий круг особенностей исходных данных. Возможно, после более глубокого исследования априорных свойств входных данных можно будет дать более точный ответ на вопрос, какой алгоритм для генерации минимального графа смежности по 35–60 вершинам стоит выбрать для конкретного набора данных.

Наконец, судя по расположению квартилей, децилей, границ доверительного интервала, можно ожидать наличия так называемых выбросов, в нашем случае — наборов данных, скорости работы на которых в двух объединенных поддиапазонах 5–60 у двух алгоритмов резко отличаются.

Медиана может быть больше или меньше 1, однако при интерпретации графиков нельзя ограничиваться лишь медианой. Необходимо посмотреть, — какой процент совокупности выше или ниже 1: медиана может быть больше 1, но 95% доверительный интервал может пересечь 1, тогда вывод о том, что один алгоритм быстрее другого, не имеет под собой основания. Для таких же целей можно рассматривать межквартильные и междецильные размахи. Они полезны в тех случаях, когда необходимо отследить «скошенность» выборок.

Так, оперируя лишь данными о среднем значении относительных времен работ двух алгоритмов, можно сделать вывод о том, что жадный алгоритм быстрее прямого в диапазоне 35–40 вершин (рис. 2), что не имеет под собой основания, ибо доверительный интервал в данном диапазоне пересекает 1, и поэтому нельзя с уверенностью говорить, что в данном диапазоне жадный алгоритм работает быстрее прямого.

Вообще говоря, на основе анализа статистических данных нельзя было бы дать однозначное заключение, что в диапазоне 35–50 вершин какой-то один алгоритм неоспоримо превосходит другой по скорости.

7. ЗАКЛЮЧЕНИЕ

В работе рассмотрены два алгоритма формирования минимального графа смежности для набора вершин (нагрузок) графа смежности. Даны описания алгоритмов, а также проведена эмпирическая оценка относительного времени выполнения алгоритмов на наборах, мощность которых находится в пределах 5–105.

Установлено, что в первом поддиапазоне 5–35 жадный алгоритм работает существенно быстрее прямого, в третьем поддиапазоне 60–105 прямой алгоритм работает существенно быстрее жадного, а во втором (среднем) поддиапазоне 35–60 имеет место переходная ситуация, а различие в скорости у алгоритмов преимущества и степени их выраженности зависят от конкретных наборов данных.

Анализ расположения значений порядковых статистик показал, что в поддиапазоне 5–60 вероятно наличие выбросов, на которых относительное время работы алгоритмов существенно отличается от среднего.

Кроме исследования прямого и жадного алгоритмов синтеза графа смежности, была выполнена разработка алгоритма, позволяющего генерировать случайные наборы входных данных — наборы нагрузок — с заданными параметрами, что позволило существенно автоматизировать проведение экспериментов.

Список литературы

1. *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.
2. *Тулупьев А.Л.* Алгебраические байесовские сети: глобальный логико-вероятностный вывод в деревьях смежности / Учеб. пособие. Сер. Элементы мягких вычислений СПб.: ООО Издательство «Анатолия», 2007. С. 40.
3. *Тулупьев А.Л.* Алгебраические байесовские сети: локальный логико-вероятностный вывод / Учеб. пособие. Сер. Элементы мягких вычислений. СПб.: ООО Издательство «Анатолия», 2007. С. 80.
4. *Тулупьев А.Л.* Алгебраические байесовские сети: открытые вопросы локального автоматического обучения // СПИСОК-2014: Материалы всероссийской научной конференции по проблемам информатики. СПб., 2014. С. 569–577.
5. *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.
6. *Тулупьев А.Л.* Дерево смежности с идеалами конъюнктов как ациклическая алгебраическая байесовская сеть // Тр. СПИИРАН. Т. 1. Вып. 3. СПб.: Наука, 2006. С. 198–227.
7. *Тулупьев А.Л., Николенко С.И., Сироткин А.В.* Байесовские сети: логико-вероятностный подход. СПб.: Наука, 2006. С. 607.
8. *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.

9. *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.
10. *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.
11. *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.))
12. *Фильченков А.А., Мусина В.Ф., Тулупьев А.Л.* Алгоритм рандомизированного синтеза минимального графа смежности // Тр. СПИИРАН. 2013. Вып. 2(35). С. 221–234.
13. *Фильченков А.А., Тулупьев А.Л.* Анализ циклов в минимальных графах смежности алгебраических байесовских сетей // Тр. СПИИРАН. 2011. №17. С. 151–173.
14. *Фильченков А.А., Тулупьев А.Л.* Структурный анализ систем минимальных графов смежности // Тр. СПИИРАН. 2009. №11. С. 104–129.
15. *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.
16. *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.
17. *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.
18. *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.
19. Maier D. Theory of Relational Databases. // Rockville, MD: Computer Science Press, 1983. P. 637
20. *Pearl J.* Causality: Models, Reasoning, and Inference. Cambridge: Cambridge University Press, 2000. P. 400.
21. Class LinkedList<E> [Electronic resource] // Java 2 Platform Standard Edition 5.0 API Specification. URL: https://docs.oracle.com/javase/1.5.0/docs/api/java/util/LinkedList.html (accessed at 28.02.2015).
22. Class template std::queue [Electronic resource] // CPlusPlus.com. URL: http://www.cplusplus.com/reference/queue/queue/ (accessed at 28.02.2015).
23. Interface Queue<E> [Electronic resource] // Java 2 Platform Standard Edition 5.0 API Specification. URL: https://docs.oracle.com/javase/1.5.0/docs/api/java/util/Queue.html (accessed at 28.02.2015).
24. Queue<T> Class [Electronic resource] // Microsoft Developer Network. URL: http://msdn.microsoft.com/ru-ru/library/7977ey2c(v=vs.110).aspx (accessed at 28.02.2015).
25. Class ArrayList<E> [Electronic resource] // Java 2 Platform Standard Edition 5.0 API Specification. URL: https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html (accessed at 28.02.2015).
26. Interface List<E> [Electronic resource] // Java 2 Platform Standard Edition 5.0 API Specification. URL: https://docs.oracle.com/javase/7/docs/api/java/util/List.html (accessed at 28.02.2015).
27. List<T> Class [Electronic resource] // Microsoft Developer Network. URL: http://msdn.microsoft.com/ru-ru/library/6sh2ey19(v=vs.110).aspx(accessed at 28.02.2015).
28. Class template std::vector [Electronic resource] // CPlusPlus.com. URL: http://www.cplusplus.com/reference/vector/vector/ (accessed at 28.02.2015).

SECONDARY STRUCTURE SYNTHESIS IN ALGEBRAIC BAYESIAN NETWORKS: A STATISTICAL TECHNIQUE FOR COMPLEXITY ESTIMATES AND COMPARATIVE ANALYSIS OF GREEDY AND STRAIGHTFORWARD ALGORITHMS

Zotov M. A., Tulupyev A. L.

Abstract

The paper considers straightforward and greedy minimal joint graph synthesis algorithms. Comparative statistical analysis of runtime was done based on experiments run on specially generated datasets. A new algorithm for generating the loads with certain characteristics was developed. Statistical analysis pointed out three subintervals of joint graph vertix set cardinality (number of elements): that of 5–35 where the greedy algorithm had sufficiently higher speed than the straightforward algorithm did, that of 60–105 the straightforward algorithm had sufficiently higher speed than the greedy algorithm did, and that of 35–60 where the algorithms advantage in their speed dependtd on each specific initial dataset. According to rank statistics, there may be detected a few autliers in the subinterval of 5–60. The paper has a didactic pupose as well.

Keywords: *uncertainty representation, algebraic Bayesian networks, probabilistic graphical models, knowledge pattern, knowledge with uncertainty, probabilistic-logic inference, statistical indicators for algorithm's complexity.*

Зотов Михаил Анатольевич,
студент 4 курса кафедры информатики математико-механического факультета СПбГУ,
zotov1994@mail.ru

Тулупьев Александр Львович,
доктор физико-математических наук, доцент, заведующий лабораторией теоретических и междисциплинарных проблем информатики СПИИРАН;
профессор кафедры информатики, математико-механический факультета СПбГУ,
alexander.tulupyev@gmail.com