НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
январь–февраль 2016 Том 16 № 1 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
January–February 2016 Vol. 16 No 1 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.8

СИНТЕЗ ВТОРИЧНОЙ СТРУКТУРЫ АЛГЕБРАИЧЕСКИХ БАЙЕСОВСКИХ СЕТЕЙ: ИНКРЕМЕНТАЛЬНЫЙ АЛГОРИТМ И СТАТИСТИЧЕСКАЯ ОЦЕНКА ЕГО СЛОЖНОСТИ

М.А. Зотов[a], Д.Г. Левенец[a], А.Л. Тулупьев[a, b], А.А. Золотин[a]

[a] Санкт-Петербургский государственный университет, Санкт-Петербург, 199178, Российская Федерация
[b] СПИИРАН, Санкт-Петербург, 199178, Российская Федерация
Адрес для переписки: Alexander.tulupyev@gmail.com

Информация о статье
Поступила в редакцию 14.12.15, принята к печати 20.12.15
doi:10.17586/2226-1494-2016-16-1-122-132
Язык статьи – русский

Ссылка для цитирования: 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.

Аннотация

Предложен улучшенный алгоритм синтеза вторичной структуры алгебраических байесовских сетей, представленной в виде минимального графа смежности. Алгоритм отличается от предложенных ранее тем, что основывается на принципе инкрементализации, использует лишь особым образом отобранные ребра, исходящие из новой вершины, исключает оставшиеся избыточные ребра с помощью жадного алгоритма. Корректность работы инкрементального алгоритма обоснована математическим доказательством. Сравнение вычислительной сложности нового (инкрементального) алгоритма и двух известных (жадного и прямого) произведено с помощью статистических оценок сложности, построенных на основе выборки значений отношения времени работы программных реализаций двух сравниваемых алгоритмов. Теоретические оценки сложности жадного и прямого алгоритмов были получены ранее, но непригодны для осуществления компаративного анализа, поскольку опираются на скрытые характеристики вторичной структуры, которые можно вычислить лишь при ее построении. Для минимизации влияния случайных факторов при вычислении отношений использовано усредненное время работы программной реализации, полученное за счет K ее запусков на одном и том же наборе нагрузок. Выборка значений отношений сформирована для M таких наборов одинаковой мощности N. По выборке вычислены среднее геометрическое со статистиками, характеризующими разброс: границы 97% доверительного интервала, а также первый и третий квартили. Приведено описание алгоритмов стохастической генерации набора нагрузок заданной мощности, а также сбора статистических данных и вычисления статистических оценок отношения времени работы прямого и жадного алгоритма ко времени работы инкрементального алгоритма. Выполнена серия экспериментов, в которых N изменяется в диапазоне 1, 2…9, 10, 26, 42… 170. Результаты серии экспериментов, визуализированные с помощью графиков с использованием библиотеки Highcharts, показали, что инкрементальных алгоритм по скорости превзошел прямой и жадный алгоритмы, причем на диапазоне мощностей наборов нагрузок 10–170 этот вывод статистически достоверен (уровень 97%). Разработанный инкрементальный алгоритм предназначен для использования в решении задач машинного обучения алгебраических байесовских сетей.

Ключевые слова

байесовские сети, синтез вторичной структуры, жадный алгоритм, инкрементальный алгоритм, вычислительная сложность, статистическая оценка, минимальный граф смежности

Благодарности

Работа 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».

SYNTHESIS OF THE SECONDARY STRUCTURE OF ALGEBRAIC BAYESIAN NETWORKS: AN INCREMENTAL ALGORITHM AND STATISTICAL ESTIMATION OF ITS COMPLEXITY

M.A. Zotov[a], D. G. Levenets[a], A.L. Tulupyev[a, b], A. A. Zolotin[a]

[a] Saint Petersburg State University, Saint Petersburg, 199178, Russian Federation
[b] SPIIRAS, Saint Petersburg, 199178, Russian Federation
Corresponding author: Alexander.tulupyev@gmail.com

Article info
Received 14.12.15, accepted 20.12.15
doi:10.17586/2226-1494-2016-16-1-122-132
Article in Russian
For citation: Zotov M.A., Levenets D.G., Tulupyev A.L., Zolotin A.A. Synthesis of the secondary structure of algebraic Bayesian networks: an incremental algorithm and statistical estimation of its complexity. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2016, vol. 16, no. 1, pp. 122–132.

Abstract
An improved algorithm for the synthesis of the secondary structure of algebraic Bayesian networks represented by a minimal join graph is proposed in the paper. The algorithm differs from the previously offered one so that it relies on the incremental principle, uses specially selected edges and, finally, eliminates redundant edges by a greedy algorithm. The correct operation of the incremental algorithm is mathematically proved. Comparison of the computational complexity of the new (incremental) algorithm implementation and two well-known (greedy and direct) is made by means of statistical estimates of complexity, based on the sample values of the runtime ratio of software implementations of two compared algorithms. Theoretical complexity estimates of the greedy and direct algorithms have been obtained earlier, but are not suitable for comparative analysis, as they are based on the hidden characteristics of the secondary structure, which can be calculated only when it is built. To minimize the influence of random factors at calculating the ratio average program runtime is used obtained by N launches on the same set of workloads. The sample values of ratio is formed for M sets of equal power K. According to the sample values the median is calculated, as well as the other statistics that characterize the spread: borders of the 97% confidence interval along with the first and the third quartiles. Sets of loads are stochastically generated according to the specified parameters using the algorithm described in the paper. The stochastic algorithms generating a set of loads with given power, as well as collecting the statistical data and calculating of statistical estimates of the ratio of forward and greedy algorithm to the incremental algorithm runtimes is described in the paper. A series of experiments is carried out in which N is changed in the range 1, 2 ... 9, 10, 26, 42 ... 170. They have showed that the incremental algorithm speed exceeds the forward and greedy ones, moreover in the 10-170 load sets power range this finding is statistically significant (97% level). The results of experiments are visualized using a graphs library Highcharts. The developed incremental algorithm is designed for application in problems solving of algebraic Bayesian networks machine learning.

Keywords
Bayesian networks, secondary structure synthesis, greedy algorithm, incremental algorithm, computational complexity, statistical estimate, minimal joint graph

Acknowledgements
The paper contains results of research partially supported with RFBR grant No.15-01-09001 – «Combined probabilistic-logic graphical approach to representation and processing of uncertain knowledge systems: algebraical Bayesian networks and related models».

Введение

Возрастающие с каждым днем объемы данных форсируют изучение методов их обработки [1], вынуждая постоянно совершенствовать подходы к преобразованию полученных «знаний». Зачастую размеры полученных данных, присутствующая в них неопределенность, связанная с нехваткой данных или их неточностью [2], а также общая связность системы в совокупности порождают проблему, для решения которой в информатике и искусственном интеллекте используется декомпозиция исходной системы на совокупность подсистем с целью локализовать вычисления, тем самым экспоненциально сократив затраты на решение поставленной задачи. Такие представители класса вероятностных графических моделей, как байесовские сети доверия, а также родственные им алгебраические байесовские сети (АБС), допускающие разбиение исходных данных на совокупности локально тесно связанных между собой объектов [3], позволяют существенно ускорить процедуру обработки новых поступающих данных (так называемых свидетельств), а также являются гибкой структурой, способной перестраиваться с учетом новых условий.

Байесовские сети доверия находят широкое применение в отрасли логистики при построении цепи поставок [4], выявлении как внутренних [5, 6], так и внешних [7] угроз и рисков для бизнеса, а также в некоторых областях исследовательской медицины [8, 9], экологических прогнозах [10, 11] и системах помощи принятия решений [12].

Однако серьезной проблемой на пути широкого применения некоторых классов вероятностных графических моделей являются необходимые затраты памяти и времени для построения их вторичной структуры или, более точно, ее машинного обучения. Например, машинное обучение байесовских сетей доверия проводится с помощью «обезьяньего поиска» [13], а также с использованием инкрементального алгоритма [14, 15], о применении, точнее, о разработке одного из шагов аналога которого в отношении АБС пойдет речь далее в настоящей работе.

Вторичная структура АБС, с одной стороны, позволяет осуществить ряд других операций машинного обучения, включая обучение параметров или локальное обучение, а с другой стороны, даже вне контекста машинного обучения эта структура используется во всех видах глобального логико-вероятностного вывода. Все эти операции возможны только тогда, когда вторичная структура АБС в определенном смысле минимальна и не содержит циклов.

Особенностью применения инкрементального алгоритма к построению вторичной структуры АБС видится то, что в каждый из k моментов времени мы переходим из предыдущего состояния с уже имеющейся построенной вторичной структурой в новое, главным отличием которого является новая вершина (фрагмент знаний), которую необходимо добавить в граф вторичной структуры. В то время как традиционные алгоритмы предоставляют единственное решение для конкретной постановки задачи, инкрементальный алгоритм позволяет адаптировать уже имеющуюся структуру и рассчитан на использование в меняющихся условиях с ограничениями на ресурсы и время [16].

В теории АБС в качестве вторичной структуры сети используется граф смежности [17–19], причем в ряде контекстов обработки и преобразования АБС требуется применение минимального графа смежности или даже дерева смежности (т.е. графа смежности без циклов). Алгоритмы синтеза минимального графа смежности по первичной структуре АБС (в данном случае достаточно говорить о корректном наборе нагрузок вершин) были предложены [20–23], однако их реализации оказались недостаточно быстрыми как для оператора, строящего вторичную структуру при последовательном внесении вершин, так и для разного рода переборных алгоритмов, которые на каждом своем шаге удаляют или вносят одну вершину в граф [19, 22, 24, 25].

Цель настоящей работы – автоматизировать синтез вторичной структуры АБС с помощью нового инкрементального алгоритма, улучшенного за счет отбора ребер, исходящих из вносимой в структуру новой вершины. Кроме того, требуется с использованием вычислительных экспериментов показать, что новый алгоритм ускоряет процесс синтеза или, иными словами, что отношение производительностей программных реализаций двух алгоритмов, описанных ранее [20, 21, 23–25], и нового алгоритма статистически значимо превосходит единицу на случайном образом сформированной выборке наборов исходных данных.

Определения, обозначения и методы

Воспользуемся системой терминов и обозначений, сложившихся в [17, 18, 22]. 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. В контексте настоящей работы вершины и их нагрузки соотносятся взаимооднозначно, поэтому мы будем использовать соответствующие им обозначения u и W_u взаимозаменяемо.

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

1. $\forall u, v \in V$ таких, что существует некоторый путь P в графе G такой, что для каждой вершины $s \in P$ справедливо утверждение $W_u \cap W_v \subseteq W_s$;
2. $\forall u, v \in V \; \langle u, v \rangle \in E$.

Граф смежности с минимальным и максимальным числом ребер мы будем называть минимальным графом смежности (МГС) и максимальным графом смежности соответственно. Путь в п. 1 определения графа смежности называется магистральным или магистралью. Граф смежности, определенный таким образом, обладает свойством магистральной связности. Пересечение нагрузок двух вершин будем называть сепаратором, а набор объектов, доступных по индексу, – списком. Под покрытием ребра понимается множество вершин графа, пересечение нагрузок которых с сепаратором этого ребра не пусто. Установлено, что система графов смежности над заданным набором нагрузок является матроидом [20, 21].

Синтез минимального графа смежности – вторичной структуры АБС – в настоящей работе описан в виде алгоритма, отличающегося тем, что, во-первых, он основывается на принципе инкрементализации, во-вторых, использует лишь особым образом отобранные ребра, исходящие из новой вершины, и, в-третьих, исключает избыточные ребра с помощью жадного алгоритма. Корректность работы инкрементального алгоритма обоснована математическим доказательством.

Теоретические оценки сложности жадного и прямого алгоритмов были получены ранее, но они непригодны для осуществления компаративного анализа [19, 22, 23]. Основанием для заключения о том, что реализация инкрементального алгоритма превосходит реализацию известных (жадного и прямого), является расположение относительно единицы доверительного интервала отношения их производительности на случайной выборке исходных данных – наборов нагрузок [24, 25]. Если весь доверительный интервал лежит выше единицы, то инкрементальный алгоритм считается более производительным, если ниже, то менее. Если доверительный интервал накрывает единицу, то такой вывод не делается, однако на основе более детального анализа возможны заключения о преобладающей тенденции.

Для минимизации влияния случайных факторов при вычислении отношений используется усредненное время работы программной реализации, полученное за счет K ее запусков на одном и том же наборе нагрузок. Выборка значений отношений формируется для M таких наборов одинаковой мощности N. По выборке вычисляются среднее геометрическое (экспонента среднего логарифмов отношений), а также статистики, характеризующие разброс: границы 97% доверительного интервала, первый и третий

квартили. Наборы нагрузок по заданным параметрам генерируются стохастически с помощью описанного в работе алгоритма.

Выполняется серия экспериментов, в которых N изменяется в определенном диапазоне с заданным шагом. Результаты серии экспериментов визуализируются с помощью графиков с использованием библиотеки Highcharts. Отметим, что в качестве меры центральной тенденции выбрано среднее геометрическое, так как именно оно участвует в построении доверительного интервала.

Стохастическая генерация наборов нагрузок

Для проведения вычислительных экспериментов, анализ результатов которых позволит судить об относительных сложностях рассматриваемых алгоритмов, требуется реализовать алгоритм стохастической генерации наборов нагрузок. Такой алгоритм был предложен в [24], но обладал следующим недостатком: не гарантировалось, что при формировании нагрузки из count элементов для очередной вершины такая нагрузка будет сформирована ровно за count шагов. В связи с этим в настоящей работе предложен новый алгоритм (листинг 1), обеспечивающий формирование нагрузки для вершины ровно за count шагов.

```
input: N, alphabet, distributions //Число генерируемых вершин, алфавит для
                                  // генерации
output: V = {v1, v2, ..., vN} //и распределения нагрузок вершин
1:  function TestSets
2:  nodes = ∅

3:  foreach (distrib in distributions) //Генерация множества нагрузок для каждого
4:     subNodes = ∅                                          //распределения
5:     countNodes = asInt(N * distrib.percent)

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

10:       while (getLength(weightOfNode) ≠ count) //Формирование нагрузки
11:          index = getRandom(0, getLength(alphabetCopy)) //для одной вершины
12:          elementWeight = alphabetCopy[index]
13:          alphabetCopy = alphabetCopy \ alphabetCopy[index]
14:          weightOfNode = weightOfNode ∪ elementWeight

15:       if (newNode(weightOfNode) not in subNodes) then // Проверка на
                                                          // уникальность
16:          subNodes = subNodes ∪ newNode(weightOfNode)

17:    nodes = nodes ∪ subNodes

18: return nodes
```

Листинг 1. Алгоритм стохастической генерации наборов нагрузок

У каждого элемента алфавита существует ненулевая вероятность попасть в нагрузку вершины. Следовательно, возможно любое сочетание нагрузок вершин, а именно это и нужно для исследования относительных сложностей алгоритмов синтеза минимального графа смежности.

Инкрементальный алгоритм синтеза минимального графа смежности

На первом шаге пополнения уже построенного минимального графа смежности G новой вершиной u требуется провести в нем новые ребра, исходящие из u в другие вершины графа, причем, с одной стороны, пополненный граф должен остаться графом смежности, а с другой стороны, среди новых ребер не должно оказаться заведомо избыточных. Для построения такого набора ребер введем функцию `GetVerticesToConnect(G,u)`(листинг 2).

```
input: G = <V,E>, v
output: V'
1:   function GetVerticesToConnect
2:   S = ∅
3:   V' = ∅
4:   H = <{V',S}> //Множество упорядоченных пар вершина-сепаратор
```

```
5:   foreach (v in V)
6:      sep = W_u ∩ W_v
7:      if(sep = ∅)
8:         continue
9:      AddAllowed = true //Уникальность вершины и соотв-щий ей сепаратор
10:     foreach (s in H.S)
11:        if(sep ⊆ s)
12:           AddAllowed = false
13:           break foreach
14:        if(sep ⊃ s)
15:           H = H \ {(v, ·)} //Удаление упорядоченной пары
16:        if(AddAllowed)
17:           H = H ∪ {(s,sep)}

18:  return H.V'
```

Листинг 2. Функция `GetVerticesToConnect`

Рассмотрим подробно строки 10–17. Вычисленный в строке 6 сепаратор сравнивается со всеми сепараторами s из H. При этом возможны три случая.

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

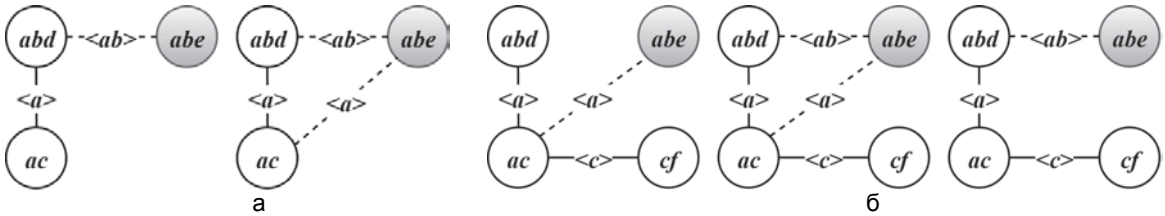

Рис. 1. Цикл с нагрузкой $\langle a \rangle$ (а); ребро $\langle a \rangle$ поглощается $\langle ab \rangle$ (б);
$a-f$ – элементы алфавита символов

2. s является подможеством сепаратора (строка 14). В этом случае новое ребро покроет собой все вершины, которые покрывались ребром с нагрузкой s, и, возможно, еще некоторые. Это означает, что после добавления нового ребра ребро с нагрузкой s станет избыточно, и его нужно удалить из H (строка 15) (см. пример на рис. 1, б).
3. Все остальные случаи (рис. 2). Тогда сепараторы либо не пересекаются, либо пересекаются частично, а значит, каждый из них покрывает свое уникальное множество вершин и не может быть поглощен другим (рис. 2, а). Следует отметить, что если пересечение сепараторов не пусто, то в граф может добавиться неэлиминируемый цикл с нагрузкой (рис. 2, б), равной их пересечению (если нагрузка ребра не между вершинами G не является подмножеством нового ребра).

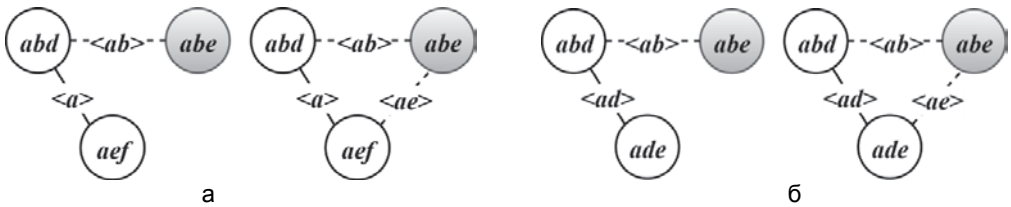

Рис. 2. Элиминируемый цикл с нагрузкой $\langle a \rangle$ (а); неэлиминируемый цикл с нагрузкой $\langle a \rangle$ (б);
$a-f$ – элементы алфавита символов

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

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

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

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

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

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

```
input: G = <V,E>, v
output: G′ = <V′,E′>
1:   function SmartIncremental
2:   E′ = E
3:   V′ = V ∪ {v}

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

6:   G′ = <V′,E′>

7:   while (true)
8:      edge = ∅

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

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

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

19:  return G′
```

Листинг 3. Алгоритм инкрементального добавления вершины в минимальный граф смежности

Описанные выше результаты использованы в алгоритме на листинге 3. Этот алгоритм инкрементален, он добавляет новую вершину в минимальный граф смежности. Ему на вход поступают уже существующий минимальный граф смежности G и новая вершина, а алгоритм возвращает минимальный граф смежности G' с добавленной новой вершиной. Алгоритм улучшен в части предварительного отбора новых ребер.

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

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

После добавления новой вершины и новых ребер формируется граф смежности G'. Далее из G' необходимо удалить все те ребра, отсутствие которых не приведет к нарушению магистральных связей (строки 7–18). Данное преобразование описано, а также улучшено в [20, 21, 24, 25]. В его результате граф G' станет минимальным графом смежности.

Дизайн эксперимента

Для сбора статистических данных, указанных в разделе «Стохастическая генерация наборов нагрузок», разработан алгоритм `Experiment` (листинг 4), который вычисляет отношение времени работы выбранного алгоритма (прямого или жадного) ко времени работы инкрементального. На вход алгоритму подаются:

- N – число нагрузок, которые требуется сгенерировать;
- `alphabet` – алфавит, из которого формируются нагрузки;
- `distributions` – распределения или правила, по которым формируются нагрузки;
- `distribution` – распределение нагрузки вершины, которая будет рассматриваться как поступающая «новая»;
- `Algorithm` – алгоритм синтеза минимального графа смежности.

```
input: N, alphabet, distributions, distribution, Algorithm
output: sortedRatios
1:  function Experiment
2:  m = 15
3:  ratioOfTimes = ∅

4:  for(i=0; i < m; i++) //Вычисления на различных наборах данных
5:     timeForAlgorithm = 0
6:     timeForIncremental = 0
7:     k = 65

8:     v = TestSets(1, alphabet, distribution) //Новая вершина
9:     V = TestSets(N, alphabet, distributions) //Множество вершин с нагрузками
10:    E = ∅
11:    G = <V,E>
12:    G = Algorithm(G) //Граф, в который будет добавляться новая вершина v

13:    for(j=0; j < k; j++) //Вычисления на одном и том же наборе данных
14:       timeForAlgorithm += getTime(Algorithm(<G.V ∪ v, ∅>))
15:       timeForIncremental += getTime(Incremental(<G.V ∪ v, G.E))

       //Подсчет среднего времени для двух алгоритмов
16:    avgTimeForSameDataAlgorithm = timeForAlgorithm / k
17:    avgTimeForSameDataIncremental = timeForIncremental / k
18:    ratioOfTimes.Add(avgTimeForSameDataAlgorithm /
       avgTimeForSameDataIncremental)

19: sortedRatios = ratioOfTimes.Sort() //Отсортированное множество отношений

20: return sortedRatios
```

Листинг 4. Алгоритм эмпирической оценки времени работы инкрементального и прямого (жадного) алгоритмов

На каждом шаге внешнего цикла (строки 4–18) формируются новые данные (строки 8–12), по которым вычисляется отношение среднего времени выполнения алгоритмов (строки 16–17). В результате будет сформирован список из отношений среднего времени выполнения алгоритмов (среднее вычисляется на одном и том же наборе данных) на различных данных. Список `ratioOfTimes` сортируется по возрастанию и подается на выход алгоритма.

Таким образом, алгоритм `Experiment` формирует результаты статистических экспериментов, проведенных для мощности в N вершин с заданным алфавитом и распределениями. Для получения таких статистик, как среднее, среднее геометрическое (экспонента среднего логарифмов отношений), медиана, первый и третий квартиль, первый и девятый дециль, верхняя и нижняя граница доверительного интервала, дисперсия, максимум и минимум, а также других остается лишь обработать соответствующим образом список, полученный на выходе алгоритма.

Дискуссия

Эксперименты были выполнены согласно описанию из предыдущего раздела и из раздела «Определения, обозначения и методы» на наборах, содержащих 1, 2, …, 9, 10, 26, 42, 58, …, 154, 170 нагрузок. Результаты экспериментов для сравнения скорости работы реализаций жадного и инкрементального алгоритмов представлены на рис. 3, для прямого и инкрементального — на рис. 4.

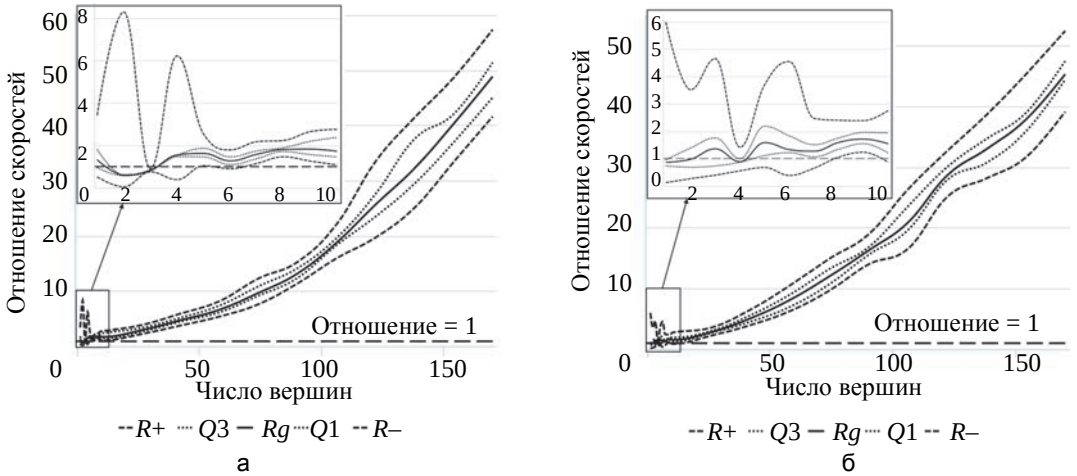

Рис. 3. Жадный и инкрементальный алгоритмы. Алфавит 48 символов. 100% вершин 3–5 порядков (а); 20% – 6–9 порядков (б)

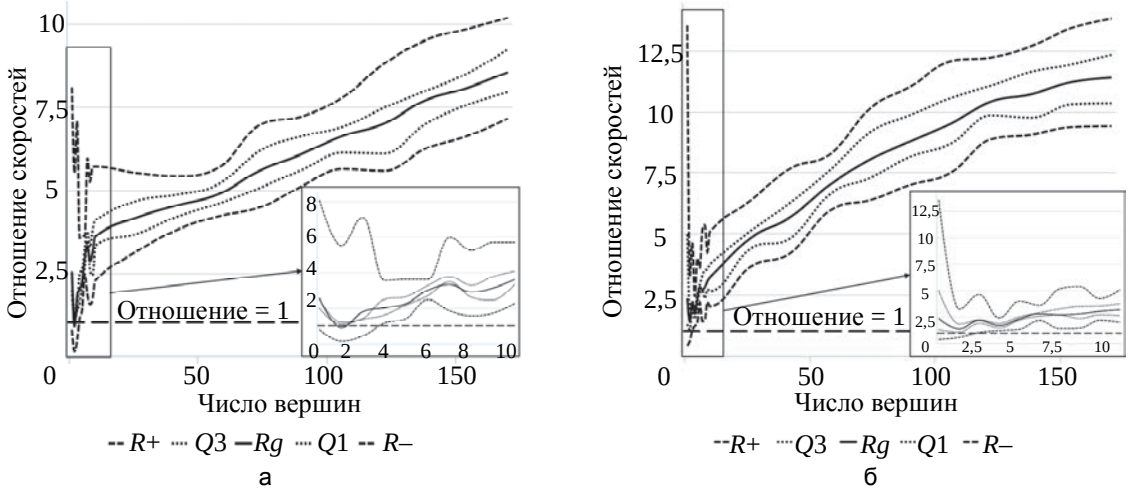

Рис. 4. Прямой и инкрементальный алгоритмы. Алфавит 48 символов. 100% вершин 3–5 порядков (а); 20% – 6–9 порядков (б)

Каждый график представляет собой визуализацию результатов статистической обработки выходных данных алгоритма `Experiment`. В качестве статистик были выбраны такие величины, как среднее геометрическое (на графиках R_g), первый и третий квартили (на графиках $Q1$ и $Q3$ соответственно), верхняя и нижняя границы доверительного интервала (на графиках R_- и R_+ соответственно). По оси абсцисс откладывалась число нагрузок в наборе (совпадает с числом вершин в синтезируемом графе), по оси ординат – отношение скоростей работы: во сколько раз прямой (жадный) алгоритм работает дольше, чем инкрементальный. Кроме того, при необходимости на графиках проведена горизонтальная прямая, параллельная оси абсцисс и имеющая значение ординаты, равное 1. Именно эта линия позволяет визуально определить, имеется ли статистически достоверное различие в скоростях алгоритмов. Анализ графиков показал, что в поддиапазоне мощностей 1–10 графики статистик плохо различимы и относительно близки, поэтому визуализация результатов из поддиапазона 1–10 размещена на масштабированных врезках.

Анализ рис. 3 позволяет сделать вывод о том, что инкрементальный алгоритм существенно превосходит в скорости работы жадный в поддиапазоне 10–170, что хорошо видно, в частности, на рис. 3, б. Для иллюстрации масштаба временных затрат отметим, что время выполнения алгоритма `Experiment` при $N = 100$ составило 222735 мс, общее время работы жадного алгоритма – 110387 мс, инкрементального – 5408 мс. В поддиапазоне 1–10 имеет место нестабильный характер отношения скоростей работы, а именно: нижняя граница доверительного интервала находится ниже 1 (на рис. 3, а), а верхняя – выше.

Таким образом, нельзя сделать статистически обоснованный вывод с заданной достоверностью о том, что на данном поддиапазоне инкрементальный алгоритм работает быстрее жадного.

Анализ рис. 4 также позволяет сделать вывод, что инкрементальный алгоритм превосходит прямой в скорости работы (см., в частности, рис. 4, б). Время выполнения алгоритма `Experiment` при $N = 100$ составило 162244 мс, общее время работы прямого алгоритма – 44458 мс, инкрементального – 5678 мс. Однако видно, что отношение скоростей начинает «расти медленнее» к мощности графа в 170 вершин, что нельзя сказать об отношении скоростей программных реализаций жадного и инкрементального алгоритмов.

Заключение

В работе предложен инкрементальный алгоритм, обеспечивающий ускоренный (по сравнению со случаем прямого и жадного алгоритмов) синтез вторичной структуры алгебраических байесовских сетей в виде минимального графа смежности. Алгоритм дополнительно улучшен тем, что производится отбор кандидатов в новые ребра: при построении связей из новой вершины берется не вся возможная совокупность исходящих из нее ребер в вершины, нагрузки которых имеют попарно непустые пересечения с нагрузкой новой, а лишь те из них, сепаратор на которых отвечает определенным условиям.

Анализ экспериментальных данных показал, что инкрементальный алгоритм на диапазоне 10–170 мощности наборов нагрузок работает в несколько раз и в десятки раз быстрее, чем прямой и жадный алгоритмы соответственно; на диапазоне 1–10 преимущество инкрементального алгоритма также имеет место, хотя и на меньшей доле элементов выборки.

Хотя в работе основное внимание уделено автоматизации построения вторичной структуры алгебраической байесовской сети, стоит отметить, что данные алгоритмы могут быть применимы также при построении графовых баз данных [22, 26], превосходящих [27] реляционные, в частности, в решении таких задач, как внесение изменений в схему базы и удовлетворение ограничений [28, 29]. В этих областях активно проводятся исследования, нацеленные на их сближение с областью применения графовых структур. Отметим, что эти исследования имеют прямое практическое применение, поскольку оптимизация запросов к базе данных [30] является немаловажным фактором работоспособности всей информационной системы в целом.

References

1. Wang X.D. Data mining in network engineering-Bayesian networks for data mining. *Proc. Int. Conf. on Education, Management, Commerce and Society*. Shenyang, China, 2015, vol. 17, pp. 412–417.
2. Benferhat S., Tabia K. Inference in possibilistic network classifiers under uncertain observations. *Annals of Mathematics and Artificial Intelligence*, 2012, vol. 64, no. 2–3, pp. 269–309. doi: 10.1007/s10472-012-9290-1
3. Stratford D.S., Croft K.M., Pollino C.A. Knowledge representation using Bayesian networks and ontologies. *Proc. 20th Int. Congress on Modelling and Simulation, ModSim 2013*. Adelaide, Australia, 2013, pp. 2980–2986.
4. Lockamy A., McCormack K. Modeling supplier risks using Bayesian networks. *Industrial Management & Data Systems*, 2012, vol. 112, no. 1–2, pp. 313–333. doi: 10.1108/02635571211204317
5. Axelrad E.T., Sticha P.J., Brdiczka O., Shen J.Q. A Bayesian network model for predicting insider threats. *Proc. 2nd IEEE CS Security and Privacy Workshops, SPW 2013*. San Francisco, USA, 2013, pp. 82–89. doi: 10.1109/SPW.2013.35
6. Hu Y., Zhang X.Z., Ngai E.W.T., Cai R.C., Liu M. Software project risk analysis using Bayesian networks with causality constraints. *Decision Support Systems*, 2013, vol. 56, pp. 439–449. doi: 10.1016/j.dss.2012.11.001
7. Xu C., Xu G.Q., Li X.H., Feng Z.Y., Meng Z.P. Approach to security attack pattern networks on the basis of Bayesian networks. *Applied Mathematics & Information Sciences*, 2013, vol. 7, pp. 233–241.
8. Wu X., Wen X.Y., Li J., Yao L. A new dynamic Bayesian network approach for determining effective connectivity from fMRI data. *Neural Computing & Applications*, 2014, vol. 24, no. 1, pp. 91–97. doi: 10.1007/s00521-013-1465-0
9. Liu K.F.R., Lu C.F., Chen C.W., Shen Y.S. Applying Bayesian belief networks to health risk assessment. *Stochastic Environmental Research and Risk Assessment*, 2012, vol. 26, no. 3, pp. 451–465. doi: 10.1007/s00477-011-0470-z
10. Forio M.A.E., Landuyt D., Bennetsen E., Lock K., Nguyen T.H.T., Ambarita M.N.D., Musonge P.L.S., Boets P., Everaert G., Dominguez-Granda L., Goethals P.L.M. Bayesian belief network models to analyse and predict ecological water quality in rivers. *Ecological Modelling*, 2015, vol. 312, pp. 222–238. doi: 10.1016/j.ecolmodel.2015.05.025

11. Murray J.V., Stokes K.E., van Klinken R.D. Predicting the potential distribution of a riparian invasive plant: the effects of changing climate, flood regimes and land-use patterns. *Global Change Biology*, 2012, vol. 18, no. 5, pp. 1738–1753. doi: 10.1111/j.1365-2486.2011.02621.x
12. Francis R.A., Guikema S.D., Henneman L. Bayesian belief networks for predicting drinking water distribution system pipe breaks. *Reliability Engineering & System Safety*, 2014, vol. 130, pp. 1–11. doi: 10.1016/j.ress.2014.04.024
13. Mittal S., Gopal K., Maskara S.L. A novel Bayesian belief network structure learning algorithm based on bio-inspired monkey search meta heuristic. *Proc. 2014 7th Int. Conf. on Contemporary Computing, IC3*. Noida, India, 2014, pp. 141–147.
14. Li S.H., Zhang J., Sun B.L., Lei J. An incremental structure learning approach for Bayesian network. *Proc. 26th Chinese Control and Decision Conference, 2014 CCDC*. Changsha, China, 2014, pp. 4817–4822.
15. Samet S., Miri A., Granger E. Incremental learning of privacy-preserving Bayesian networks. *Applied Soft Computing*, 2013, vol. 13, no. 8, pp. 3657–3667. doi: 10.1016/j.asoc.2013.03.011
16. Sharp A.M. *Incremental Algorithms: Solving Problems in a Changing World*. PhD Dissertation. Cornell University, USA, 2007, 121 p. Available at: http://www.cs.cornell.edu/~asharp/papers/thesis.pdf.
17. Tulup'ev A.L. *Algebraicheskie Baiesovskie Seti: Global'nyi Logiko-Veroyatnostnyi Vyvod v Derev'yakh Smezhnosti* [Algebraical Bayesian Networks: Global Logic-Probabilistic Inference in the Adjacent Tree]. St. Petersburg, SPbSU Publ., Anatolia Publ., 2007, 40 p.
18. Tulup'ev A.L., Sirotkin A.V., Nikolenko S.I. *Baiesovskie Seti Doveriya: Logiko-Veroyatnostnyi Vyvod v Atsiklicheskikh Napravlennykh Grafakh* [Bayesian Belief Networks: Logical-Probabilistic Inference in the Acyclic Directed Graph]. St. Petersburg, SPbSU Publ., 2009, 400 p.
19. Tulupyev A.L., Stolyarov D.M., Mentyukov M.V. A representation for local and global structures of an algebraical Bayesian network in Java applications. *SPIIRAS Proceedings*, 2007, no. 5, pp. 71–99.
20. Oparin V.V., Tulupyev A.L. Synthesis of a joint graph with minimal number of edges: an algorithm formalization and a proof of correctness. *SPIIRAS Proceedings*, 2009, no. 11, pp. 142–157.
21. Oparin V.V., Filchenkov A.A., Sirotkin A.V., Tulupyev A.L. Matroidal representation for the adjacency graphs family built on a set of knowledge patterns. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2010, no. 4 (68), pp. 73–76.
22. Filchenkov A.A. *Sintez Grafov Smezhnosti v Mashinnom Obuchenii Global'nykh Struktur Algebraicheskikh Baiesovskikh Setei. Dis. ... kand. fiz.-mat. nauk* [Synthesis Join Graph in Machine Learning of Global Structures of Algebraic Bayesian Networks. Dis. Phys.-Math. Sci.]. Samara, 2013, 339 p.
23. Filchenkov A.A., Tulupyev A.L., Sirotkin A.V. Algebraic Bayesian network minimal join graphs: formalization of synthesis basis and automated learning. *Fuzzy Systems and Soft Computing*, 2011, vol. 6, no. 2, pp. 145–163.
24. Zotov M.A., Tulupyev A.L Secondary structure synthesis in algebraic Bayesian networks: a statistical technique for complexity estimates and comparative analysis of greedy and straightforward algorithms. *Computer Tools in Education*, 2015, no. 1, pp. 3–18.
25. Zotov M.A., Tulupyev A.L., Sirotkin A.V. Complexity statistical estimates of straightforward and greedy algorithms for algebraic Bayesian networks syntesis. *Fuzzy Systems and Soft Computing*, 2015, vol. 10, no. 1, pp. 75–91.
26. Konstantinou N., Spanos D.E., Kouis D., Mitrou N. An approach for the incremental export of relational databases into RDF graphs. *International Journal on Artificial Intelligence Tools*, 2015, vol. 24, no. 2, art. 1540013. doi: 10.1142/S0218213015400138
27. Wycislik L., Warchal L. A performance comparison of several common computation tasks used in social network analysis performed on graph and relational databases. *Man-Machine Interactions 3*, 2014, vol. 242, pp. 651–659. doi: 10.1007/978-3-319-02309-0_70
28. Gao J., Zhou J.S., Yu J.X., Wang T.J. Shortest path computing in relational DBMSs. *IEEE Transactions on Knowledge and Data Engineering*, 2014, vol. 26, no. 4, pp. 997–1011. doi: 10.1109/TKDE.2013.43
29. Chen R.W. Managing massive graphs in relational DBMS. *Proc. IEEE Int. Conf. on Big Data*. Santa Clara, 2013, pp. 1–8.
30. Park J., Lee S.G. A graph-theoretic approach to optimize keyword queries in relational databases. *Knowledge and Information Systems*, 2014, vol. 41, no. 3, pp. 843–870. doi: 10.1007/s10115-013-0690-2

Зотов Михаил Анатольевич – студент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199178, Российская Федерация, zotov1994@mail.ru

Левенец Даниил Григорьевич – студент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199178, Российская Федерация, daniel-levenets@yandex.ru

Тулупьев Александр Львович – доктор физико-математических наук, доцент, заведующий лабораторией, СПИИРАН, Санкт-Петербург, 199178, Российская

Федерация; профессор, Санкт-Петербургский государственный университет, Санкт-Петербург, 199178, Российская Федерация, Alexander.tulupyev@gmail.com

Золотин Андрей Алексеевич – студент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199178, Российская Федерация, andrey.zolotin@gmail.com

Mikhail A. Zotov – student, Saint Petersburg State University, Saint Petersburg, 199178, Russian Federation, zotov1994@mail.ru

Daniil G. Levenets – student, Saint Petersburg State University, Saint Petersburg, 199178, Russian Federation, daniel-levenets@yandex.ru

Alexander L Tulupyev – D.Sc., Associate professor, Head of Laboratory, SPIIRAS, Saint Petersburg, 199178, Russian Federation; Professor, Saint Petersburg State University, Saint Petersburg, 199178, Russian Federation, Alexander.tulupyev@gmail.com

Andrey A. Zolotin – student, Saint Petersburg State University, Saint Petersburg, 199178, Russian Federation, andrey.zolotin@gmail.com