НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
сентябрь–октябрь 2016 Том 16 № 5 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
September–October 2016 Vol. 16 No 5 ISSN 2226-1494 http://ntv.ifmo.ru/en
УНИВЕРСИТЕТ ИТМО

УДК 004.8

СИНТЕЗ ЧЕТВЕРТИЧНОЙ СТРУКТУРЫ АЛГЕБРАИЧЕСКИХ БАЙЕСОВСКИХ СЕТЕЙ: ИНКРЕМЕНТАЛЬНЫЙ И ДЕКРЕМЕНТАЛЬНЫЙ АЛГОРИТМЫ

А.В. Романов[a,b], А.А. Золотин[a], А.Л. Тулупьев[a, b]

[a] Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация
[b] СПИИРАН, Санкт-Петербург, 199178, Российская Федерация
Адрес для переписки: Alexander.tulupyev@gmail.com

Информация о статье
Поступила в редакцию 16.06.16, принята к печати 20.07.16
doi: 10.17586/2226-1494-2016-16-5-917-927
Язык статьи – русский

Ссылка для цитирования: Романов А.В., Золотин А.А., Тулупьев А.Л. Синтез четвертичной структуры алгебраических байесовских сетей: инкрементальный и декрементальный алгоритмы // Научно-технический вестник информационных технологий, механики и оптики. 2016. Т. 16. № 5. С. 917–927. doi: 10.17586/2226-1494-2016-16-5-917-927

Аннотация

Предмет исследования. Алгебраические байесовские сети относятся к классу вероятностных графических моделей, являющихся представлением баз знаний с неопределенностью. Отличительной особенностью алгебраических байесовских сетей является наличие глобальных структур, среди которых присутствуют первичная и вторичная, которые используются в различных видах логико-вероятностного вывода непосредственно, а также третичная и четвертичная, участвующие в задачах автоматического синтеза, идентификации свойств вторичной структуры и, опосредованно, в решении задач машинного обучения указанных сетей. Существующие алгоритмы изменения четвертичной структуры требуют ее полного перестроения при изменении первичной структуры. Эта особенность замедляет синтез глобальных структур, рассеивает внимание пользователя, вынужденного заново анализировать всю перестроенную структуру, а не концентрироваться лишь на тех изменениях, которые были непосредственно обусловлены ограниченной модификацией исходных данных, что снижает привлекательность алгебраических байесовских сетей как модели для обработки и визуализации данных в целом. **Цель исследования**. Работа направлена на локализацию изменений четвертичной структуры при добавлении и удалении вершины из первичной структуры или, иначе говоря, на обеспечение ограничения таких изменений лишь необходимыми из них за счет устранения недостатка существующих алгоритмов, выраженного в избыточном перестроении всей структуры. Для достижения поставленной цели решается задача инкрементализации алгоритма перестроения четвертичной структуры. **Метод.** Предлагаемый подход основан на свойствах инкрементализации алгоритмов, позволяющих сократить объем вычислений за счет использования результата, полученного на предыдущем шаге алгоритма. Все рассуждения, проводимые в работе, изложены на языке теории графов, чтобы использовать устоявшуюся систему терминов и классических результатов. **Основные результаты.** В работе представлены инкрементальный и декрементальный алгоритмы изменения четвертичной структуры при добавлении или удалении вершины из первичной структуры алгебраической байесовской сети, снабженные доказательством корректности и листингами. Приведенные в работе алгоритмы строятся на основании полученных ранее инкрементальных алгоритмов для третичной структуры. Также в работе проводится подробный анализ множества сепараторов на каждом этапе работы алгоритмов. **Теоретическая и практическая значимость**. Полученные алгоритмы развивают теорию глобальных структур алгебраических байесовских сетей, в частности, и теорию вероятностных графических моделей в целом. Кроме того, они создают задел для поиска инвариантного представления системы связей внутри алгебраической байесовской сети в отличие от существующего подхода, при котором эти связи выражены во вторичной структуре, которая может оказаться не единственной даже при фиксированной первичной структуре. Исключение манипуляций с множеством вторичных структур заметно упростит и сделает обозримой визуализацию такого сложного объекта, как алгебраические байесовские сети, и, возможно, улучшит вычислительные характеристики алгоритмов логико-вероятностного вывода в сети, а также позволит переформулировать задачи машинного обучения алгебраических байесовских сетей, исключив из них необходимость синтеза множества разных объектов сложной структуры с фактически одинаковой семантикой. Также можно ожидать, что полученные инкрементальные алгоритмы ускорят вычислительные процессы по перестроению и анализу свойств всех четырех глобальных структур алгебраических байесовских сетей.

Ключевые слова
алгебраические байесовские сети, синтез четвертичной структуры, инкрементальный алгоритм, машинное обучение, структурный синтез, вероятностные графические модели.

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».

QUATERNARY STRUCTURE SYNTHESIS IN ALGEBRAIC BAYESIAN NETWORKS: INCREMENTAL AND DECREMENTAL ALGORITHMS

A.V. Romanov[a,b], A.A. Zolotin[a], A.L. Tulupyev[a, b]

[a] Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation
[b] SPIIRAS, Saint Petersburg, 199178, Russian Federation
Corresponding author: Alexander.tulupyev@gmail.com

Article info
Received 16.06.16, accepted 20.07.16
doi: 10.17586/2226-1494-2016-16-5-917-927
Article in Russian
For citation: Romanov A.V., Zolotin A.A., Tulupyev A.L. Quaternary structure synthesis in algebraic bayesian networks: incremental and decremental algorithms. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2016, vol. 16, no. 5, pp. 917–927. doi: 10.17586/2226-1494-2016-16-5-917-927

Abstract
Subject of Research. Algebraic Bayesian networks are referred to a class of probabilistic graphical models that are a representation of knowledge bases with uncertainty. The distinguishing feature of ABN is the availability of global structures. Among them there are primary and secondary structures that are directly used in various kinds of probabilistic logical inference as well as tertiary and quaternary involved in the problems of automatic synthesis and identification of the properties of the secondary structure and partially in the machine learning tasks within specified networks. Existing algorithms for quaternary structure changing require its complete rebuild when changing the primary structure. That feature slows down the whole global structures synthesis, dispels user's attention who is forced to re-analyze the whole rebuilt structure instead of focusing on the changes that were directly caused by the limited modification of the original data. This fact reduces ABN attractiveness as a model for data processing in general. **Scope of Research.** This paper is aimed at speeding up the rebuild process and eliminating the shortage of the quaternary structure rebuild algorithms when adding and deleting vertices of primary structure expressed in excess rebuild of the entire structure. The task of algorithm incrementalization for quaternary structure rebuild is solved to achieve the goal. **Method.** The proposed approach is based on the properties of incremental algorithms that reduce the amount of computations due to the result obtained at the previous step of the algorithm. All the arguments used in the paper are expressed in a graph theory language to apply the established system of terms and classical results. **Main Results.** The paper presents incremental and decremental algorithms, complemented by a proof of correctness and listing. Given algorithms are based on the previously obtained incremental algorithms for tertiary structure. Moreover, a detailed analysis of the plurality of separators is carried out on each stage of the algorithms. **Theoretical and Practical Relevance.** These algorithms develop a global structure of algebraic Bayesian networks as well as the theory of probabilistic graphical models in general. Furthermore, they create the groundwork for creation of the secondary structure invariants that may be non-unique even if the primary structure is fixed unlike the current approach where connections are included in the secondary structure. The deletion of manipulation with a set of secondary structures considerably simplifies and makes foreseeable visualization of such complex object as ABN and may improve the computational characteristics of probabilistic logical inference algorithms. It also enables to reformulate the problem of ABN machine learning excluding any need for synthesis of many different objects with complex structure and virtually the same semantics. It also can be expected that the obtained incremental algorithms will accelerate computational processes of rebuilding and properties analysis for all four global ABN structures.

Keywords
algebraic Bayesian networks, quaternary structure synthesis, incremental algorithm, machine learning, structural synthesis, probabilistic graphical models

Acknowledgements
The paper presents results of the project partially supported by the RFBR grant 15-01-09001-a "Combined probabilistic-Logic Graphical Approach to Representation and Processing of Uncertain Knowledge Systems: Algebraical Bayesian Networks and Related Models"

Введение

Алгебраические байесовские сети (АБС) являются одним из представлений баз знаний с неопределенностью [1–3]. Они родственны байесовским сетям доверия, широко используемым в оценке рисков [4, 5], системах поддержки принятия решений [6] и прогнозировании [7]. Классическим представлением АБС является граф, в вершинах которого стоят фрагменты знаний, являющиеся совокупностями достаточно тесно связанных объектов, которым приписаны оценки вероятностей, а ребра представляют логические связи между вершинами, обладающими общими элементами [3, 8]. Одной из особенностей байесовских сетей является наличие как локальной структуры (фрагмента знаний), так и множества глобальных структур, среди которых – первичная, вторичная, третичная и четвертичная структуры [8, 9]. Первичная и вторичная структуры необходимы для проведения глобального вывода, в то время как третичная и четвертичная структуры служат другим целям: выявлению ацикличности вторичной структуры [10, 11], построению всего множества минимальных графов смежности [12], а также поиску наиболее эффективной вторичной структуры [13]. Существующие алгоритмы перестроения четвертичной структу-

ры при изменении первичной структуры АБС основываются на полном перестроении, что является избыточным, замедляет синтез глобальных структур, рассеивает внимание пользователя, вынужденного заново анализировать всю перестроенную структуру, а не концентрироваться лишь на тех изменениях, которые были непосредственно обусловлены ограниченной модификацией исходных данных, что снижает привлекательность АБС как модели для обработки и визуализации данных в целом. Взаимосвязанность структур, а также размеры моделей при больших объемах данных существенно усложняют полное перестроение каждой из структур при изменении количества вершин, а при небольших изменениях в модели (добавление или удаление нескольких вершин) возможность «на лету» (just-in-time) перестроить модель позволяет ускорить обработку данных.

Частной целью данной работы является упрощение отслеживания изменений четвертичной структуры, которые обусловлены ограниченным изменением первичной структуры, т.е. внесением одной новой вершины в набор вершин либо исключению из него одной вершины. Такое упрощение достигается за счет «локализации» изменений, выполнения тех изменений, которые требуются, а не вообще всех возможных, как это происходит при построении четвертичной структуры «заново». Локализация изменений четвертичной структуры достигается за счет анализа изменений, произошедших в третичной структуре АБС. С вычислительной точки зрения достижение цели сводится к решению одной из задач структурного синтеза, а именно к инкрементализации уже известного алгоритма синтеза четвертичной структуры [12].

Инкрементальный подход, на котором основывается предложенный в настоящей работе алгоритм, также используется в теории графов [14, 15], статистике [16], задачах классификации [17, 18]. Применение указанного подхода к модификации алгоритма синтеза четвертичной структуры АБС основывается на решении задач перестроения третичной структуры согласно алгоритмам, предложенным в [19, 20], и последующем анализе множества сепараторов. Полученные инкрементальный и декрементальный алгоритмы дополняются листингами и доказательством корректности.

Отметим, что достижение частной цели, возможно, вносит вклад в достижение еще одной цели исследований – ускорение синтеза четвертичной структуры. Однако в силу особенностей теоретических оценок сложности соответствующих алгоритмов [21] их компаративный анализ сводится к проведению правильно организованной серии вычислительных экспериментов и статистической обработке их результатов по аналогии с [8, 22], что формирует содержание отдельного исследования и поэтому не рассматривается в настоящей работе.

Определения, обозначения и методы

Чтобы сформировать систему обозначений, приведем определения и понятия, уже сложившиеся в [2, 8, 12].

Графом $G = \langle V, E \rangle$ называется совокупность двух множеств – непустого множества V (множества вершин) и множества E двухэлементных подмножеств множества V (E – множество ребер). Нагруженный граф – тройка (G, A, W), где G – граф; A – алфавит атомов; W – функция нагрузки, заданная на множествах вершин и ребер G со значениями из 2^A.

Сепаратором двух вершин в нагруженном графе назовем пересечение нагрузок соответствующих вершин. В контексте данной работы будем для удобства отождествлять понятие вершины v и нагрузки вершины W_v, заменяя одно на другое. Также будем рассматривать только согласованные графы, т.е. такие, у которых нагрузка любого ребра равна пересечению нагрузок вершин этого ребра.

Максимальный граф смежности $G_{\max}$ – наибольший по числу ребер граф смежности. Сужение максимального графа смежности G на сепаратор u – это подграф графа G, в который входят только те вершины и ребра, нагрузки которых содержат или равны u.

Третичная структура – это граф, представленный в виде диаграммы Хассе с порядком следования сверху вниз, построенный на множестве непустых сепараторов (пример на рис. 1, а).

Родителем сепаратора будем называть сепаратор, предшествующий данному в третичной структуре. Сыном сепаратора будем называть сепаратор, следующий за данным в третичной структуре.

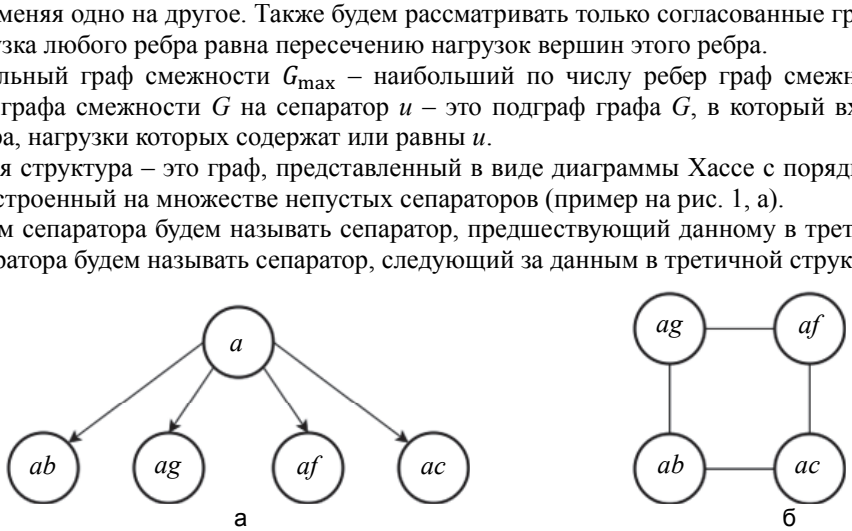

Рис. 1. Пример третичной (а) и четвертичной (б) структуры алгебраической байесовской сети. В узлах графов указаны нагрузки-сепараторы, состоящие из одного или нескольких атомов алфавита.

Полусиблинговый граф сепаратора *u* – это граф, построенный над множеством его сыновей, ребро между двумя вершинами которого проведено тогда и только тогда, когда сужения, соответствующие данным вершинам, пересекаются.

Другими словами, сужения максимального графа смежности будут пересекаться только в том случае, если пересекаются множества вершин данных сужений. Такое возможно, только если более одной вершины принадлежат обоим сужениям, а тогда такие вершины содержат в себе оба сепаратора, по которым строятся сужения.

Получается, что сужения двух сыновей сепаратора *u* пересекаются тогда, когда в максимальном графе смежности присутствует вершина, нагрузка которой содержит в себе оба сына.

Семейство полусиблинговых графов, построенных для каждого элемента множества сепараторов, которое пополнено пустым сепаратором, является четвертичной структурой (пример на рис. 1, б).

Инкрементальное добавление вершины в четвертичную структуру

Рассмотрим изменение четвертичной структуры при добавлении новой вершины в АБС. Так как четвертичная структура строится по третичной, первое, что надо сделать – это соответствующим образом перестроить третичную структуру. Алгоритм изменения третичной структуры при добавлении вершины в АБС представлен в [20]. Но, так как четвертичная структура представляет собой семейство полусиблинговых графов, построенных для каждого сепаратора из третичной структуры, то для ее изменения понадобятся следующие данные.

1. Все новые сепараторы, чтобы добавить построенные по множеству их сыновей полусиблинговые графы в четвертичную структуру – NewSeps.
2. Все вершины из третичной структуры, у которых изменилось множество сыновей, так как это отразится на соответствующем данной вершине полусиблинговом графе – Parents.
3. Все сепараторы, которые входят в новую вершину и уже содержатся в третичной структуре, так как могут появиться новые ребра в уже построенном полусиблинговом графе, содержащем данные сепараторы – OldSeps.

Если же при добавлении новой вершины третичная структура не изменяется, т.е. не появляется новых сепараторов, то в четвертичной структуре при этом изменения могут проявиться только в виде новых ребер в отдельных полусиблинговых графах.

Будем возвращать, кроме перестроенной третичной структуры, три множества: Parents, содержащее все вершины из третичной структуры, у которых поменялось множество сыновей, NewSeps, содержащее все новые сепараторы, добавленные в третичную структуру в процессе работы алгоритма, и OldSeps, содержащее все сепараторы, которые входят в новую вершину и в третичную структуру, добавленные в третичную структуру в процессе работы алгоритма.

Таким образом, мы получим все сепараторы, полусиблинговые графы которых изменятся при добавлении новой вершины в АБС, а также все сепараторы, между которыми могут появиться ребра. Для изменения четвертичной структуры останется перестроить полусиблинговые графы для полученных множеств сепараторов Parents и NewSeps и проверить на добавление новых ребер сепараторы из OldSeps, у которых будет общий родитель в третичной структуре.

Описание инкрементального алгоритма

В листинге 1 представлен псевдокод изменения четвертичной структуры при добавлении новой вершины.

На вход алгоритму подается построенная четвертичная структура GraphsCollection, представленная в виде коллекции полусиблинговых графов, набор вершин *V*, необходимый для функции изменения третичной структуры, добавляемая вершина *u* и третичная структура, представленная графом *G*. На выходе получаем измененную четвертичную структуру, представленную в виде коллекции полусиблинговых графов GraphCollection.

Сперва происходит вызов функции AddSeparatorsFourth (строка 4), которая вызывает алгоритм изменения третичной структуры при добавлении новой вершины и возвращает множество родителей Parents, у которых изменилось множество сыновей в процессе работы функции, и множество NewSeps, содержащее новые сепараторы, добавленные в третичную структуру, а также множество OldSeps, в котором содержатся сепараторы, входящие в новую вершину и в третичную структуру. После этого необходимо перестроить полусиблинговые графы для каждого сепаратора, множество сыновей которого изменилось (строки 6–17), построить полусиблинговые графы для новых сепараторов (строки 18–26) и проверить добавление новых ребер между сепараторами из множества OldSeps (строки 27–32).

В процессе перебора элементов множества Parent создаем ссылку на граф, соответствующий текущему сепаратору *p* (строка 7), получаем множество сыновей для данного сепаратора из измененной третичной структуры (строка 8). Далее удаляем из графа вершины и ребра, содержащие сепараторы, которые уже не являются сыновьями *p* (строки 9–11). После этого добавим в граф все новые сыновья

(строки 13–14) и соединим ребрами добавленную вершину с остальными вершинами графа, если сужения на них максимального графа смежности пересекаются (строки 15–17). Далее необходимо добавить в коллекцию полусиблинговых графов новые графы для каждого нового сепаратора из третичной структуры, т.е. из множества NewSeps. Переберем все элементы множества NewSeps (строка 18), в процессе перебора для каждого текущего сепаратора *s* добавим новый граф в коллекцию (строка 19), создадим ссылку на этот граф (строка 18), получим множество сыновей для *s* (строка 21).

Переберем и добавим все элементы множества сыновей sons во множество вершин текущего графа currentGraph (строка 23). Останется только соединить ребрами те вершины графа, для которых сужения на данные сепараторы пересекаются, что и происходит в строках 24–26. После этого переберем все элементы множества OldSeps (строки 27–28), получим родителя сепараторов из третичной структуры, либо null, если его нет (строка 29), и, если есть общий родитель, между текущими сепараторами нет ребра в полусиблинговом графе их родителя и их сужения пересекаются (строки 30–32), то добавим новое ребро (строка 33) в полусиблинговый граф родителя.

Далее пояснены некоторые идентификаторы, используемые в листингах алгоритмов:
currentGraph – полусиблинговый граф из четвертичной структуры для текущего сепаратора;
SonsFromThird (*p*, *G*) – получаем коллекцию сыновей сепаратора *p* из третичной структуры *G*;
IsConstrictionIntersect (*s*, *v*) – проверяет, пересекаются ли сужения сепараторов *s* и *v*, возвращает true, если пересекаются;
GetParent (*m*, *n*) – возвращает родителя сепараторов *m* и *n* из третичной структуры, null – если общего родителя нет.

```
input: GraphCollection, V, u, G=(S,E)
output: GraphCollection
1: functionRebuildQuaternaryStructureInc
2: Parents = ∅
3: NewSeps = ∅
4: OldSeps = ∅
5: AddSeparatoursFourth(G, V, u, Parents, NewSeps, OldSeps)
6: foreach (p in Parents)
7:       currentGraph = GraphCollection[p]
8:       sons = SonsFromThird(p, G)
9:       foreach (v incurrentGraph.V)
10:                if (v not in sons)
11:                      currentGraph.Remove(v)
12:           foreach (s in sons)
13:                if (s not incurrentGraph.V)
14:                      currentGraph.AddVertex(s)
15:                      foreach (v incurrentGraph.V)
16:                            if (IsConstrictionIntersect(s,v)
                               and v ≠ s)
17:                                  currentGraph.AddEdge(s,v)
18:      foreach (s inNewSeps)
19:           GraphCollections.AddNewGraph(s,NewGraph)
20:           currentGraph = GraphsCollection[s]
21:           sons = SonsFromThird(s,G)
22:           foreach (son in sons)
23:                currentGraph.AddVertex(son)
24:                foreach (v incurrentGraph.V)
25:                      if (IsConstrictionIntersect(v,s) and v ≠ s)
26:                            currentGraph.AddEdge(v,s)
27:      for (i = 0; I <OldSeps.Count - 1; i.Inc)
28:           for (j = i + 1; j <OldSeps.Count; j.Inc)
29:                parent = GetParent(OldSeps[i],OldSeps[j])
30:                if (parent ≠ null andGraphCollection[parent].
                   Edges.Contains({OldSeps[i],OldSeps[j]}))
                   andIsConstrictionIntersect(OldSeps[i],OldSeps[j]))
31:                      GraphCollection[parent].
                         AddEdge({OldSeps[i],OldSeps[j]})
32:returnGraphCollection
```

Листинг 1. Инкрементальный алгоритм изменения четвертичной структуры при добавлении новой вершины

Обоснование корректности инкрементального алгоритма

Теорема 1. Пусть заданы первичная структура АБС V и соответствующие ей третичная структура в виде графа $G = \langle S, E \rangle$ и четвертичная структура в виде коллекции полусиблинговых графов GraphsCollection. При добавлении одной новой вершины u в V, функция RebuildQuaternaryStructureInc изменяет коллекцию GraphsCollection так, что GraphsCollection будет являться четвертичной структурой для АБС, построенной на множестве вершин $V \cup \{u\}$.

Доказательство. Четвертичная структура представляет собой семейство полусиблинговых графов, построенных для каждого сепаратора из третичной структуры, а при появлении новой вершины в третичной структуре могут появиться новые сепараторы, что повлечет за собой появление новых полусиблинговых графов, либо новых сепараторов не будет, но в любом случае могут измениться связи между старыми. В ходе доказательства поделим условно третичную структуру на три множества, и для каждого из указанных множеств представим алгоритм обработки изменений, произошедших в нем в результате добавления вершины. Таким образом, чтобы проверить, что полученная структура будет являться четвертичной, необходимо проанализировать действия над тремя множествами, составляющими третичную структуру: множество измененных сепараторов третичной структуры, множество новых сепараторов третичной структуры и множество неизмененных сепараторов. Над каждым элементов из вышеуказанных множеств проводятся следующие операции:

1. для каждого сепаратора из третичной структуры, у которого поменялось множество сыновей, перестроить соответствующий ему полусиблинговый граф из четвертичной структуры;
2. для каждого нового сепаратора из третичной структуры, появившегося после добавления новой вершины, построить полусиблинговый граф и добавить его в четвертичную структуру;
3. для всех старых сепараторов, которые уже входили в третичную структуру, проверить условия для проведения между ними ребра в полусиблинговом графе их родителя в случае отсутствия данного ребра.

Продемонстрируем, что после работы алгоритма с листинга 1 все вышеперечисленные действия будут выполнены.

Покажем, что первый пункт списка выполняется. После вызова функции AddSeparatorsFourth в переменной Parents будут лежать все сепараторы из третичной структуры, множество сыновей у которых поменялось. Далее в строках 6–17 выделяется полусиблинговый граф, соответствующий текущему сепаратору p из Parents, в котором удаляются вершины и ребра, связанные с этими вершинами, которые уже не содержатся в текущем множестве сыновей для p из третичной структуры, а следовательно, не должны содержаться в полусиблинговом графе для данного сепаратора (строки 9–11). Далее для всех новых сепараторов s из множества сыновей sons сепаратора p, которые не содержатся во множестве вершин полусиблингового графа, по нему построенного, выполняются следующие действия: s добавляется во множество вершин (строка 14), затем проводятся ребра между s и остальными сепараторами из множества вершин графа в соответствии с определением полусиблингового графа (строки 15–17). Таким образом, для каждого сепаратора из третичной структуры, у которого изменилось множество сыновей, перестроился соответствующим образом полусиблинговый граф, т.е. пункт 1 выполнен.

Выполнению второго пункта списка соответствуют строки 18–26, в которых рассматриваются все новые сепараторы s (строка 18), создается и добавляется в четвертичную структуру новый полусиблинговый граф для s (строки 19–20), который заполняется вершинами и ребрами в соответствии с определением полусиблингового графа (строки 23–25), все сыновья добавляются во множество вершин (строка 23), и проводятся ребра между теми вершинами, сужения на которые максимального графа смежности пересекаются (строки 24–26). Таким образом, условие 2 выполнено, а значит, получившееся семейство полусиблинговых графов является четвертичной структурой для нового набора вершин.

Третий пункт списка выполняется в строках 27–33. Происходит перебор всех элементов множества OldSeps (строки 27–29), проверяется наличие общего родителя (строка 29), и, если родитель существует, ребро между двумя сепараторами еще не проведено, но условия для проведения ребра из определения полусиблингового графа выполнены (строки 30–32), то добавляется новое ребро (строка 33).

Согласно вышеприведенному алгоритму, область данных операций покрывает собой все три множества, на которые нами была условно разделена третичная структура, таким образом, проводится проверка всех изменений, произошедших в ней, а значит, поскольку четвертичная структура строится над третичной, получившееся в результате семейство графов будет являться четвертичной структурой для нового набора вершин. Таким образом, алгоритм корректен и формирует четвертичную структуру для указанной АБС, что и требовалось доказать. ■

Декрементальное удаление вершины из четвертичной структуры

Рассмотрим изменение четвертичной структуры при удалении вершины из АБС. Ввиду того, что четвертичная структура строится по третичной, сперва необходимо соответствующим образом изменить третичную структуру с помощью алгоритмов, описанных выше.

Четвертичная структура представляет собой семейство полусиблинговых графов, построенных по сепараторам из третичной структуры, у которых есть сыновья. Таким образом, изменения в четвертичной структуре могут быть в трех случаях:

1. если изменятся какие-либо полусиблинговые графы, что возможно, когда у соответствующих сепараторов в третичной структуре поменяется набор сыновей;
2. если изменится само семейство графов, т.е. из него удалятся некоторые графы, что произойдет, когда удалятся соответствующие им сепараторы из третичной структуры;
3. если у сепараторов в полусиблинговом перестанут пересекаться сужения после удаления вершины. Тогда ребро между ними необходимо будет удалить.

Таким образом, для изменения четвертичной структуры понадобится получить после работы алгоритма изменения третичной структуры дополнительные множества: набор сепараторов, которые были удалены из третичной структуры, набор вершин, у которых поменялось множество сыновей, и множество всех сепараторов из третичной структуры, которые входят в удаляемую вершину.

Будем возвращать множество удаляемых сепараторов DeletedSeps, сепараторы Parents, у которых изменилось множество сыновей, и множество OldSeps сепараторов, которые входят в удаляемую вершину. При каждом удалении сепаратора из третичной структуры будем возвращать множество его родителей, тогда после работы алгоритма AddSeparatorsFourth в множестве Parents будут находиться все сепараторы, у которых был удален сын, что привело к изменению их множества сыновей. Так как в данном случае никаким другим способом множество сыновей у сепаратора поменяться не может, то в итоге получим необходимое множество.

Описание декрементального алгоритма

В листинге 2 представлен псевдокод алгоритма изменения четвертичной структуры при удалении вершины.

```
input: GraphsCollection, V, u, Seps, G=(S,E)
output: GraphsCollection
1: functionRebuildQuaternaryStructureDec
2: Parents = ∅
3: DeletedSeps = ∅
4: OldSeps = ∅
5: RemoveSeparatorsFourth(G,V,u,Seps,DeletedSeps,Parents,OldSeps)
6: foreach (d inDeletedSeps)
7:    GraphCollection.RemoveGraph(d)
8: foreach (p in Parents)
9:    currentGraph = GraphCollection[p]
10:   sons = SonsFromThird(p, G)
11:   foreach (v incurrentGraph.V)
12:         if (v not in sons)
13:               currentGraph.Remove(v)
14:   foreach (s in sons)
15:         if (s not incurrentGraph.V)
16:               currentGraph.AddVertex(s)
17:               foreach (v incurrentGraph.V)
18:                     if (IsConstrictionIntersect(s, v) and v ≠ s)
19:                           currentGraph.AddEdge(s, v)
20: foreach (o1 inOldSeps)
21:   foreach (o2 inOldSeps)
22:         p = FindParent(o1,o2)
23:         if ((o1 != o2)and(p != null) and
            (!IsConstrictionIntersect(o1,o2)))
24:                     GraphsCollection[p].RemoveEdge((o1,o2))
25: returnGraphCollection
```

Листинг 2. Декрементальный алгоритм изменения четвертичной структуры при исключении вершины

На вход алгоритму подаются четвертичная структура, представленная семейством графов GraphCollection, множество вершин V, удаляемая вершина u, множество сепараторов Sep, третичная структура, представленная графом G. На выходе получаем измененную четвертичную структуру, соответствующую новому набору вершин.

Сначала необходимо изменить третичную структуру и получить множества сепараторов, полусиблинговые графы которых будут изменены (строка 4). Затем удалим из семейства графов все полусиблин-

говые графы (строки 5–6), сепараторы которых были удалены из третичной структуры в процессе работы функции RemoveSeparatorsFourth. После этого переберем все сепараторы Parents, множество сыновей которых изменилось, и перестроим соответствующие им полусиблинговые графы (строки 7–18). В процессе перебора для каждого сепаратора p из Parents удалим вершины из соответствующего ему полусиблингового графа currentGraph, которые уже не содержатся во множестве сыновей p (строки 9–12).

Далее найдем среди сыновей сепараторы, которые еще не принадлежат множеству вершин currentGraph.V (строки 13–14), добавим их во множества вершин (строка 15), после чего соединим добавленные сепараторы с остальными вершинами в соответствии с определением полусиблингового графа (строки 16–18).

Осталось проверить все сепараторы из OldSeps, которые были родителями удаленных из третичной структуры сепараторов. Если между сепараторами o1 и o2 из OldSeps было проведено ребро в полусиблинговом графе, а теперь их сужения не пересекаются (строки 20–24), то данное ребро необходимо удалить (строка 24).

Обоснование корректности декрементального алгоритма

Теорема 2. Пусть заданы первичная структура АБС V и соответствующие ей третичная структура в виде графа $G = \langle S, E \rangle$ и четвертичная структура в виде коллекции полусиблинговых графов GraphsCollection. При исключении одной вершины u из V функция RebuildQuaternaryStructureInc изменяет коллекцию GraphsCollection так, что GraphsCollection будет являться четвертичной структурой для АБС, построенной на множестве вершин $V \setminus \{u\}$.

Доказательство. При удалении вершины четвертичная структура меняется тогда и только тогда, когда меняется третичная. Если меняется третичная, значит, в нее добавляются новые сепараторы, либо в нашем случае удаляются. Так как четвертичная структура представляет собой семейство полусиблинговых графов, построенных для каждого сепаратора из третичной структуры, то нас интересуют два вида сепараторов из третичной структуры после ее изменения:

1. удаленные сепараторы;
2. сепараторы, у которых изменилось множество сыновей.

Если сепаратор не попадает ни под один из пунктов, то его полусиблинговый граф останется неизменным, значит, нужно только изменить для всех сепараторов, подходящих под один из пунктов, соответствующие им полусиблинговые графы, чтобы четвертичная структура соответствовала новому набору вершин. Рассмотрим оба случая.

1. Для удаленных сепараторов необходимо удалить из семейства их полусиблинговые графы, что и происходит в строках 5–6.
2. Для сепараторов, у которых изменилось множество сыновей, необходимо перестроить соответствующий им полусиблинговый граф (строки 7–18), а именно, удалить все вершины, которые не входят в новое множество сыновей текущего сепаратора (строки 9–12), и добавить вершины, которые входят во множество сыновей, но еще не добавлены во множество вершин, а также соединить их ребрами с другими вершинами по правилам в соответствии с определением полусиблингового графа.

Кроме того, между соединенными ребром сепараторами, у которых был удален общий сын из третичной структуры и сужения которых теперь не пересекаются, ребра быть не должно. Проверка данного условия всех сепараторов с изменившимся множеством сыновей производится в строках 20–24: сначала определяется, есть ли у них общий родитель (строка 22), затем, если сужения данных сепараторов больше не пересекаются (строка 23), то ребро между ними исключается из полусиблингового графа.

После всех вышеперечисленных действий для всех вершин из третичной структуры, которые подходили под одно из двух условий, были изменены соответствующие им полусиблинговые графы в четвертичной структуре. Таким образом, мы рассмотрели все изменения, произошедшие в третичной структуре при удалении вершины, а значит, поскольку четвертичная структура строится над третичной, то полученная в результате работы алгоритма структура является четвертичной для нового набора вершин, что и требовалось доказать. ■

Пример

Приведем пример использования алгоритмов. Пусть задана первичная структура АБС {*ac*, *abg*, *aeg*, *afg*, *adf*, *adq*}. Пример соответствующей ей вторичной структуры изображен на рис. 2. Как видно из рисунка, граф содержит три различных сепаратора: *a*, *ag* и *af*.

На рис. 3 представлены третичная (рис. 3, а) и четвертичная структура (рис. 3, б). Четвертичная представляет собой один полусиблинговый граф для сепаратора *ac*, состоящий их сепараторов *ag* и *af*, причем между ними проведено ребро, так как существует вершина *afg*, содержащая оба сепаратора. Для *ag* и *af* графы будут пустыми, так как у данных сепараторов нет сыновей в третичной структуре.

Если добавить в первичную структуру АБС вершину *adq*, то вторичная структура изменится, как показано на рис. 4. Таким образом, в третичной структуре, а также в полусиблинговом графе сепаратора *a* появится новая вершина *ad* (рис. 5).

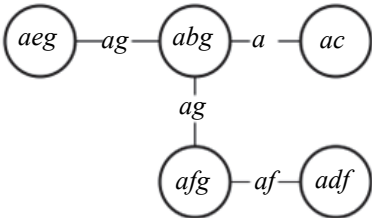

Рис. 2. Вторичная структура

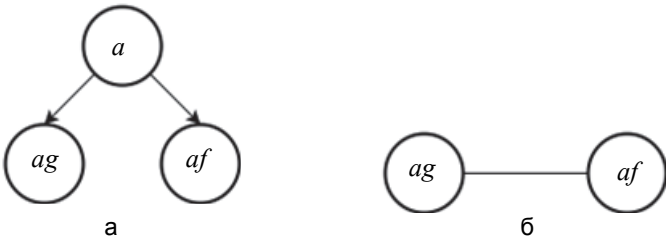

Рис. 3. Третичная (а) и четвертичная (б) структуры

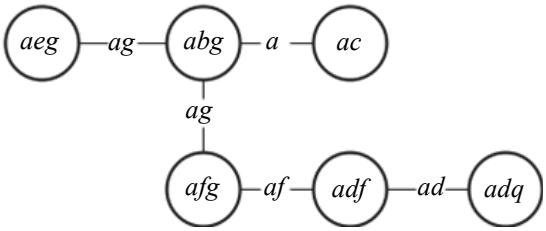

Рис. 4. Вторичная структура с новой вершиной *adq*

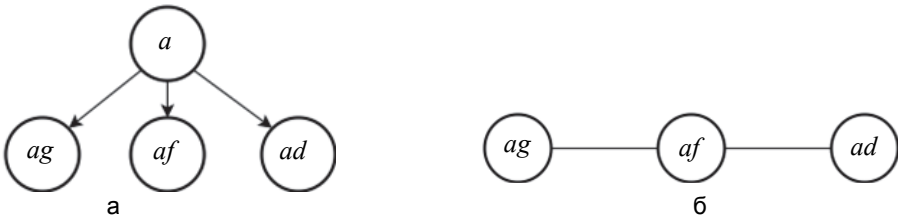

Рис. 5. Третичная (а) и четвертичная (б) структуры после добавления вершины *adq*

Если теперь удалить из первичной структуры вершину *afg*, то вторичная структура примет вид, как показано на рис. 6. В третичной структуре останется 3 вершины, а полусиблинговый граф сепаратора *a* будет содержать две несвязные вершины *ag* и *ad*, так как теперь нет вершины, в которую бы входили оба данных сепаратора (рис. 7).

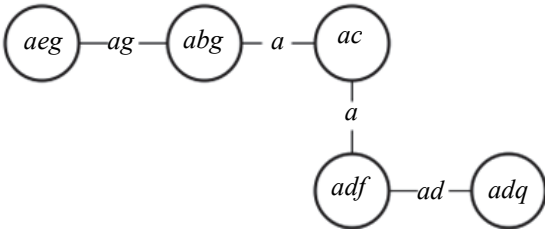

Рис. 6. Вторичная структура после исключения вершины *afg*

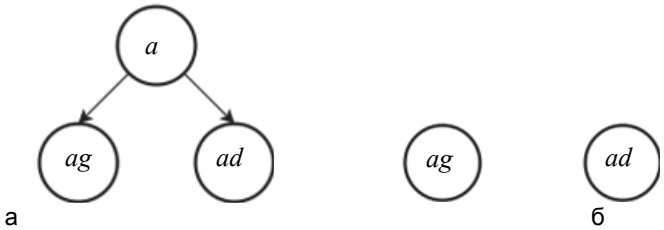

Рис. 7. Третичная (а) и четвертичная (б) структуры после удаления вершины *afg*

Заключение

Рассмотренный в работе метод формирования четвертичной структуры алгебраической байесовской сети является принципиально новым. Предполагается, что в совокупности все предложенные результаты сделают анализ четвертичной структуры и сам алгоритм ее перестроения более обозримым за счет минимизации изменений в структуре и рассмотрения из всех сепараторов третичной структуры лишь множества, указанного в предложенном алгоритме.

Кроме того, есть основания ожидать, что полученные результаты позволят ускорить синтез четвертичной структуры алгебраической байесовской сети за счет принятия во внимание текущего состояния системы, что должно дать преимущество по сравнению с полным перестроением сети. Такое ускорение наблюдается в родственной ситуации – при инкрементализации синтеза вторичной структуры [8].

Тем самым полученные результаты развивают теорию алгебраических байесовских сетей в целом и теорию глобальных структур этих сетей, в частности. Кроме того, развитие инкрементальных и декрементальных алгоритмов над третичной и четвертичной структурами создает задел для развития альтернатив использованию вторичной структуры в виде минимального графа смежности для логико-вероятностного вывода в алгебраических байесовских сетях. Такие альтернативы могут оказаться инвариантными, в отличие от минимальных графов смежности, которые могут быть необозримо многочисленными при одной и той же первичной структуре алгебраической байесовской сети.

Литература

1. Тулупьев А.Л., Николенко С.И., Сироткин А.В. Байесовские сети доверия: логико-вероятностный подход. СПб.: Наука, 2006. 607 с.
2. Тулупьев А.Л., Сироткин А.В., Николенко С.И. Байесовские сети доверия: логико-вероятностный вывод в ациклических направленных графах. СПб.: СПбГУ, 2009. 400 с.
3. Тулупьев А.Л. Алгебраические байесовские сети: локальный логико-вероятностный вывод: Учеб. пособие. СПб.: СПбГУ-Анатолия, 2007. 80 с.
4. Kabir G., Sadiq R., Tesfamariam S. A fuzzy Bayesian belief network for safety assessment of oil and gas pipelines // Structure and Infrastructure Engineering. 2016. V. 12. N 8. P. 874–889. doi: 10.1080/15732479.2015.1053093
5. John A., Yang Z., Riahi R., Wang J. A risk assessment approach to improve the resilience of a seaport system using Bayesian networks // Ocean Engineering. 2016. V. 111. P. 136–147. doi: 10.1016/j.oceaneng.2015.10.048
6. Zarikas V., Papageorgiou E., Regner P. Bayesian network construction using a fuzzy rule based approach for medical decision support // Expert Systems. V. 32. N 3. P. 344–369. doi: 10.1111/exsy.12089
7. Zhang Q. Dynamic uncertain causality graph for knowledge representation and reasoning: continuous variable, uncertain evidence, and failure forecast // IEEE Transactions on Systems Man, and Cybernetics: Systems. 2015. V. 45. N 7. P. 990–1003. doi: 10.1109/TSMC.2015.2392711
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. doi: 10.17586/2226-1494-2016-16-1-122-132
9. Banerjee S., Ghosal S. Bayesian structure learning in graphical models // Journal of Multivariate Analysis. 2015. V. 136. P. 147–162. doi: 10.1016/j.jmva.2015.01.015
10. Фильченков А.А., Тулупьев А.Л. Анализ циклов в минимальных графах смежности алгебраических байесовских сетей // Труды СПИИРАН. 2011. № 2. С. 151–173.
11. Фильченков А.А., Тулупьев А.Л. Алгоритм выявления ацикличности первичной структуры алгебраической байесовской сети по ее четвертичной структуре // Труды СПИИРАН. 2011. № 4(19). С. 128–145.
12. Фильченков А.А. Синтез графов смежности в машинном обучении глобальных структур алгебраических байесовских сетей. Дис. ... канд. физ.-мат. наук. Самара, 2013. 339 с.
13. Фильченков А.А., Тулупьев А.Л. Третичная структура алгебраической байесовской сети // Труды СПИИРАН.

References

1. Tulup'ev A.L., Nikolenko S.I., Sirotkin A.V. *Baiesovskie Seti: Logiko-Veroyatnostnyi Podkhod* [Bayesian Networks: Logical-Probabilistic Approach]. St. Petersburg, Nauka Publ., 2006, 607 p.
2. 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.
3. Tulup'ev A.L. *Algebraicheskie Baiesovskie Seti: Lokal'nyi Logiko-Veroyatnostnyi Vyvod* [Algebraic Bayesian Networks: a Local Logic-Probabilistic Inference]. St. Petersburg, SPbGU Publ., Anatoliya, 2007, 80 p.
4. Kabir G., Sadiq R., Tesfamariam S. A fuzzy Bayesian belief network for safety assessment of oil and gas pipelines. *Structure and Infrastructure Engineering*, 2016, vol. 12, no. 8, pp. 874–889. doi: 10.1080/15732479.2015.1053093
5. John A., Yang Z., Riahi R., Wang J. A risk assessment approach to improve the resilience of a seaport system using Bayesian networks. *Ocean Engineering*, 2016, vol. 111, pp. 136–147. doi: 10.1016/j.oceaneng.2015.10.048
6. Zarikas V., Papageorgiou E., Regner P. Bayesian network construction using a fuzzy rule based approach for medical decision support. *Expert Systems*, vol. 32, no. 3, pp. 344–369. doi: 10.1111/exsy.12089
7. Zhang Q. Dynamic uncertain causality graph for knowledge representation and reasoning: continuous variable, uncertain evidence, and failure forecast. *IEEE Transactions on Systems Man, and Cybernetics: Systems*, 2015, vol. 45, no. 7, pp. 990–1003. doi: 10.1109/TSMC.2015.2392711
8. 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. doi: 10.17586/2226-1494-2016-16-1-122-132
9. Banerjee S., Ghosal S. Bayesian structure learning in graphical models. *Journal of Multivariate Analysis*, 2015, vol. 136, pp. 147–162. doi: 10.1016/j.jmva.2015.01.015
10. Filchenkov A.A., Tulupyev A.L. The algebraic Bayesian network minimal join graphs cycles analysis. *SPIIRAS Proceedings*, 2011, no. 2, pp. 151–173.
11. Filchenkov A.A., Tulupyev A.L. Algorithm for detection of algebraic Bayesian network primary structure acyclicity based on its quaternary structure. *SPIIRAS Proceedings*, 2011, no. 4, pp. 128–145.
12. Filchenkov A.A. *Sintez Grafov Smezhnosti v Mashinnom Obuchenii Global'nykh Struktur Algebraicheskikh Baiesovskikh Setei. Dis. ... kand. fiz.-mat. nauk* [Synthesis Join Graph in

2011. № 3(18). С. 164–187.
14. Bernstein A. Maintaining shortest paths under deletions in weighted directed graphs // SIAM Journal on Computing. 2016. V. 45. N 2. P. 548–574. doi: 10.1137/130938670
15. Soulignac F.J. Fully dynamic recognition of proper circular-arc graphs // Algorithmica. 2015. V. 71. N 4. P. 904–968. doi: 10.1007/s00453-013-9835-7
16. Lu G.-F., Zou J., Wang Y. A new and fast implementation of orthogonal LDA algorithm and its incremental extension // Neural Processing Letters. 2016. V. 43. N 3. P. 687–707. doi: 10.1007/s11063-015-9441-6
17. Feng L., Wang Y., Zuo W. Quick online spam classification method based on active and incremental learning // Journal of Intelligent and Fuzzy Systems. 2016. V. 30. N 1. P 17–27. doi: 10.3233/IFS-151707
18. Chen M.-H., Chang P.-C., Wu J.-L. A population-based incremental learning approach with artificial immune system for network intrusion detection // Engineering Applications of Artificial Intelligence. 2016. V. 51. P. 171–181. doi: 10.1016/j.engappai.2016.01.020
19. 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 // Proc. 1st Int. Scientific Conference on Intelligent Information Technologies for Industry (IITI'16). Sochi, Russia, 2016. P. 57–67. doi: 10.1007/978-3-319-33816-3_6
20. Романов А.В., Левенец Д.Г., Золотин А.А., Тулупьев А.Л. Инкрементальный синтез третичной структуры алгебраических байесовских сетей // Сборник докладов Международной конференции по мягким вычислениям и измерениям (SCM-2016). Санкт-Петербург, 2016. Т. 1. С. 27–29.
21. 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. 3–18.
22. Зотов М.А., Тулупьев А.Л., Сироткин А.В. 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.

Machine Learning of Global Structures of Algebraic Bayesian Networks. Dis. Phys.-Math. Sci.]. Samara, 2013, 339 p.
13. Filchenkov A.A., Tulupyev A.L. The Algebraic Bayesian Network Tertiary Structure. *SPIIRAS Proceedings*, 2011, no. 3, pp. 164–187.
14. Bernstein A. Maintaining shortest paths under deletions in weighted directed graphs. *SIAM Journal on Computing*, 2016, vol. 45, no. 2, pp. 548–574. doi: 10.1137/130938670
15. Soulignac F. J. Fully dynamic recognition of proper circular-arc graphs. *Algorithmica*, 2015, vol. 71, no. 4, pp. 904–968. doi: 10.1007/s00453-013-9835-7
16. Lu G.-F., Zou J., Wang Y. A new and fast implementation of orthogonal LDA algorithm and its incremental extension. *Neural Processing Letters*, 2016, vol. 43, no. 3, pp. 687–707. doi: 10.1007/s11063-015-9441-6
17. Feng L., Wang Y., Zuo W. Quick online spam classification method based on active and incremental learning. *Journal of Intelligent and Fuzzy Systems*, 2016, vol. 30, no. 1, pp. 17–27. doi: 0.3233/IFS-151707
18. Chen M.-H., Chang P.-C., Wu J.-L. A population-based incremental learning approach with artificial immune system for network intrusion detection. *Engineering Applications of Artificial Intelligence*, 2016, vol. 51, pp. 171–181. doi: 10.1016/j.engappai.2016.01.020
19. 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. *Proc. 1st Int. Scientific Conference on Intelligent Information Technologies for Industry, IITI'16*. Sochi, Russia, 2016, pp. 57–67. doi: 10.1007/978-3-319-33816-3_6
20. Romanov A.V., Levenets D.G., Zolotin A.A., Tulup'ev A.L. Incremental synthesis of tertiary structure of algebraic Bayesian networks. *Proc. Int. Conf. on Soft Computing and Measurements*. St. Petersburg, 2016, vol. 1, pp. 27–29. (In Russian)
21. 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.
22. 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.

Авторы

Романов Артем Витальевич **–** студент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация; младший научный сотрудник, СПИИРАН, Санкт-Петербург, 199178, Российская Федерация, exxemka.spb@gmail.com

Золотин Андрей Алексеевич **–** аспирант, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, andrey.zolotin@gmail.com

Тулупьев Александр Львович **–** доктор физико-математических наук, профессор, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация; доцент, заведующий лабораторией, СПИИРАН, Санкт-Петербург, 199178, Российская Федерация, Alexander.tulupyev@gmail.com

Authors

Artem V. Romanov **–** student, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation; junior scientific researcher, SPIIRAS, Saint Petersburg, 199178, Russian Federation, exxemka.spb@gmail.com

Andrey A. Zolotin **–** postgraduate, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation, andrey.zolotin@gmail.com

Alexander L. Tulupyev **–** D.Sc., Professor, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation; Associate professor, Head of Laboratory, SPIIRAS, Saint Petersburg, 199178, Russian Federation, Alexander.tulupyev@gmail.com