УДК 004.8 **DOI: 10.14529/cmse230104**

ПРИМЕНЕНИЕ ТРЕТИЧНОЙ СТРУКТУРЫ АЛГЕБРАИЧЕСКОЙ БАЙЕСОВСКОЙ СЕТИ В ЗАДАЧЕ АПОСТЕРИОРНОГО ВЫВОДА

© 2023 А.А. Вяткин[1], М.В. Абрамов[1], Н.А. Харитонов[2], А.Л. Тулупьев[3]

[1] *Санкт-Петербургский федеральный исследовательский центр Российской академии наук (199178 Санкт-Петербург, 14-я лин. В.О., д. 39),*

[2] *Санкт-Петербургский государственный университет (199034 Санкт-Петербург, Университетская набережная, д. 7-9),*

[3] *Северо-Западный институт управления Российской академии народного хозяйства и государственной службы при Президенте Российской Федерации (199034 Санкт-Петербург, Средний пр. В.О., д. 57/43)*

E-mail: aav@dscs.pro, mva@dscs.pro, nak@dscs.pro, alt@dscs.pro

Поступила в редакцию: 09.12.2022

В теории алгебраических байесовских сетей существуют алгоритмы, позволяющие проводить глобальный апостериорный вывод с использованием вторичных структур. При этом построение вторичных структур предполагает использование третичной структуры. Следовательно, возникает вопрос об обособленном применении третичной структуры в задаче апостериорного вывода. Этот вопрос рассматривался ранее, но было приведено только общее описание алгоритма, при этом учитывались лишь модели со скалярными оценками вероятности истинности. В данной работе приведен алгоритм, расширяющий вышеупомянутый до возможности его использования в случае интервальных оценок. Помимо этого, важным свойством алгебраической байесовской сети является ацикличность, и корректность работы перечисленных алгоритмов обеспечивается только для ациклических сетей. Поэтому необходимо также уметь проверять ацикличность алгебраической байесовской сети с применением третичной структуры. Описание этого алгоритма также представлено в работе, в его основе лежит ранее доказанная теорема, которая связывает количество моделей фрагментов знаний в сети с количеством непустых сепараторов и количеством компонент связности сильных сужений в цикличной АБС, а также доказанная в данной статье теорема о принадлежности двух моделей фрагментов знаний к одной компоненте связности сильного сужения. Для всех разработанных алгоритмов доказана корректность работы, а также вычислена их оценка временной сложности.

Ключевые слова: алгебраические байесовские сети, фрагмент знаний, логико-вероятностный вывод, третичная структура, вероятностные графические модели, машинное обучение.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Вяткин А.А., Абрамов М.В., Харитонов Н.А., Тулупьев А.Л. Применение третичной структуры алгебраической байесовской сети в задаче апостериорного вывода // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 1. С. 61–88. DOI: 10.14529/cmse230104.

Введение

Вероятностные графические модели сегодня находят широкое применение в самых разных областях науки, технологий и промышленности [1]. Например, они используются в задачах распознавания и отслеживания людей на видео [2], анализа кредитного риска [3], исследования влияния человеческого фактора на морские аварии [4], оценки вероятности успеха многоходовых социоинженерных атак [5, 6]. Один из важных представителей класса вероятностных графических моделей — алгебраические байесовские сети [7]. В своей простейшей — первичной — структуре они являются набором идеалов конъюнктов со ска-

лярными или интервальными оценками вероятности истинности (набор моделей фрагментов знаний) и применяются в целях описания экспертных систем [7]. Для более широкого практического использования алгебраических байесовских сетей необходимо развить соответствующий теоретический аппарат через решение ряда задач, одной из которых посвящен данный материал.

В контексте теории алгебраических байесовских сетей рассматриваются различные вопросы, в частности, — распространение (пропагация) свидетельства, то есть пересчет оценок вероятностей истинности элементов алгебраической байесовской сети на основе свидетельства, интерпретацией которого является новая, ранее неизвестная информация о предметной области. Например, если имеется сформированная модель, описывающая прогноз погоды, то новой информацией будет утверждение о том, что завтра определенно пойдет снег. Распространение (пропагация) свидетельства является задачей глобального апостериорного вывода. Более формально, свидетельство — либо предположение о том, что какие-то утверждения оказались истинными или ложными, либо о том, что над подыдеалом, описывающем часть утверждений, задана модель фрагмента знаний с новыми оценками вероятности истинности. Для пропагации свидетельства могут использоваться вторичная или третичная структуры. На данный момент для решения этой задачи применяется механизм распространения виртуальных свидетельств во вторичной структуре алгебраической байесовской сети [8]. Вторичная структура является графом смежности, где вершины нагружены математическими моделями фрагментов знаний (идеалы конъюнктов со скалярными или интервальными оценками вероятности истинности) [8]. Виртуальным же свидетельством называют пересечение двух смежных вершин вторичной структуры, также являющееся моделью фрагмента знаний, при этом пересечение задается между парами вершин, где в первой вершине уже учтена информация, поступившая от изначального свидетельства, а во второй — нет. Вторичная структура для заданной первичной не определена однозначно, поэтому необходимо отдельно исследовать способы построения вторичных структур, в которых распространение свидетельства происходило бы наименее ресурсозатратно. Для синтеза таких вторичных структур, то есть их построения на основе первичных с, возможно, дополнительными ограничениями на вид графа, в настоящий момент используются третичные структуры [9, 10]. Третичная структура — это диаграмма Хассе над замкнутым множеством всех непустых пересечений пар моделей фрагментов знаний [11].

Иными словами, для пропагации свидетельств нужна вторичная структура, для построения которой используется третичная структура. Отсюда возникает вопрос: есть ли возможность производить апостериорный вывод, применяя только третичную структуру, без построения вторичной, что позволило бы уменьшить количество применяемых объектов и, возможно, ускорить общую работу модели? Таким образом, исследование новых алгоритмов апостериорного вывода, в частности с применением только третичной структуры, является актуальным, поскольку позволило бы достичь обозначенных выше результатов: уменьшить количество применяемых объектов и, вероятно, ускорить общую работу модели. Этот вопрос ранее рассматривался в [12], но только для скалярных оценок вероятности истинности, при этом алгоритм описывался в общих чертах, без явного представления псевдокода.

Следует также отметить, что еще важным свойством алгебраической байесовской сети является ацикличность. Ее проверка необходима, так как только для ацикличных сетей доказана корректность вышеупомянутого алгоритма, наличие такого свойства позволяет

значительно уменьшить вычислительную сложность поддержания непротиворечивости алгебраической байесовской сети, что также важно при использовании ее на практике. Таким образом, целью работы является расширение и анализ алгоритма апостериорного вывода, применяющего третичную структуру, до возможности его использования в случае интервальных оценок. В связи с этой целью выделяются следующие задачи:

1) описать расширение рассматриваемого в [12] алгоритма апостериорного вывода;
2) сформировать алгоритм проверки ацикличности алгебраической байесовской сети с применением третичной структуры, что ранее не проводилось;
3) доказать корректность работы этих алгоритмов;
4) оценить их сложность.

Теоретическая значимость работы заключается в формировании возможности дальнейшего применения полученных результатов в исследованиях задачи апостериорного вывода в алгебраических байесовских сетях, исследовании третичной структуры как самодостаточного объекта, необходимого при практическом применении алгебраических байесовских сетей. Практическая значимость заключается в возможном ускорении работы алгоритмов апостериорного вывода за счет уменьшения количества создаваемых объектов и расширении за счет этого области применения данного аппарата.

Опишем краткое содержание разделов данной статьи. В разделе 1 рассматриваются работы, в которых исследовались вопросы, связанные с изучаемыми в данной статье проблемами, а также работы, на которые опирается данная статья. Раздел 2 знакомит читателя с теоретической основой, необходимой для понимания дальнейшего изложения статьи. В разделе 3 описывается апостериорный вывод, применяющий третичную структуру. Раздел 4 посвящен алгоритму проверки ацикличности алгебраической байесовской сети. В разделе 5 кратко описываются и анализируются полученные результаты — работа алгоритмов, их сложность. Заключение подводит итоги исследования и обозначает направления дальнейших работ.

1. Релевантные работы

Помимо алгебраических байесовских сетей существуют другие классы вероятностных графических моделей, которые позволяют решать те же задачи. Наиболее распространенными являются байесовские сети доверия (БСД). БСД находят широкое применение в различных областях. Так, они используются в анализе безопасности и надежности систем, оценке рисков [13–18], обработке естественного языка [19], изучении настроения пользователей социальной сети [20], повышении безопасности сотрудников на рабочих местах [21], разработке городской инфраструктуры [22], здравоохранении [23], климатологии [24], анализе аварий при судоходстве [25], в сетях газопровода [26]. При этом в БСД используются только точечные оценки вероятности истинности, а также остаются открытыми вопросы, связанные с обработкой направленных циклов [27]. В теории АБС же могут учитываться интервальные оценки вероятности истинности, что позволяет, например, легче формализовывать неопределенность высказывания на естественном языке, учитывать наблюдения с частично пропущенными или утраченными данными [27]. Помимо этого в теории АБС существуют подходы, позволяющие использовать циклические структуры [28].

Вернемся к рассмотрению задачи построения алгоритма апостериорного вывода. Ранее использовался механизм распространения виртуальных свидетельств [8], использующий вторичную структуру, построение которой описано, например, в [29]. Данная же рабо-

та рассматривает использование только третичной структуры при проведении глобального апостериорного вывода. Фундаментом работы послужило исследование [12], в котором описывается апостериорный вывод в алгебраической байесовской сети с использованием только третичной структуры в случае скалярных оценок. Эта статья является на текущий момент единственной, посвященной подобному вопросу. В ней представлено общее описание алгоритма вывода, а также доказана его корректность для ациклических алгебраических байесовских сетей. Для применения этого алгоритма на практике требуется доработка, чтобы иметь возможность использовать его в алгебраических байесовских сетях с интервальными оценками вероятности истинности. Данная статья посвящена доработке указанного алгоритма, а также оценке его сложности, что необходимо для дальнейшего практического применения и анализа его работы.

Второй основной вопрос, которому посвящена статья — проверка ацикличности алгебраической байесовской сети. Такую проверку необходимо осуществлять, так как только для ациклических байесовских сетей доказана корректность работы вышеупомянутого алгоритма, и потому свойство ацикличности важно для корректной работы расширяющего алгоритма, описываемого в данной работе. Эта проблема рассматривалась в [30], но задача решалась с применением, опять же, новых структур (четвертичных), помимо третичных. При этом, так как предполагается использование только третичной структуры для практического применения алгебраических байесовских сетей, проверку следует осуществлять, не создавая новые структуры, то есть использовать лишь третичную структуру. Работ, посвященных задаче проверки ацикличности в такой постановке, нет. Но стоит отметить, что решение этой задачи в текущей статье опирается на теорему о цикличности первичной структуры, описанную в [31].

2. Теоретическая основа

В данной главе опишем систему терминов и ряд алгоритмов, используемых в данной работе, на которые будет опираться дальнейшее изложение материала. Данный раздел основан на более ранних работах [8, 11, 12, 27, 32–34].

2.1. Основные определения

Прежде всего рассмотрим объекты, которые будут соответствовать переменным, заключающим утверждения. Они образуют *алфавит* — множество, состоящее из атомарных пропозициональных формул (которые могут называться атомами) [27]. $A = \{x_1, ..., x_n\}$ определяет алфавит из n атомов. Для оценки самих атомарных пропозиций, а также связей между ними определим *идеал конъюнктов*, построенный над алфавитом $A = \{x_1, ..., x_n\}$ — множество формул вида $\{x_{i_1}x_{i_2}...x_{i_k} \mid 0 \leq i_1 < ... < i_k \leq n-1, k \leq n\}$, где $x_{i_1}x_{i_2}...x_{i_k}$ представляет конъюнкцию соответствующих переменных [27].

Рассмотренным выше атомам, а также пропозициональным формулам и элементам идеала конъюнктов в частности сопоставим оценки вероятности истинности, которые характеризуют степень уверенности в утверждениях и наборах утверждений, соответствующих атомам и пропозициональным формулам. Таким образом определим объект, расширяющий идеал конъюнктов дополнительными свойствами — *математическую модель фрагмента знаний* (для краткости обозначим как ФЗ) [27]. Математической моделью фрагмента знаний, который построен над алфавитом A, назовем пару (C, p), где C — идеал конъюнктов

над соответствующим алфавитом, p — интервальные или скалярные (точечные) оценки вероятностей для каждого конъюнкта из идеала C [27].

Оценки вероятности истинности для каждого конъюнкта из идеала C, соответствующего некоторому ФЗ, могут быть противоречивыми. Формализуем это свойство. Допустим задан ФЗ (C, p) со скалярными оценками вероятности истинности соответствующих конъюнктов. Тогда этот ФЗ будет *непротиворечив* в том и только том случае, если существует заданная на множестве конъюнктов из ФЗ вероятность p_f, такая что $\forall c \in C, p_f(c) = p(c)$ [33]. Если задан ФЗ с интервальными оценками $(C, \mathbf{p})$, то он будет непротиворечив тогда и только тогда, когда $\forall c \in C$ и $\forall \alpha \in \mathbf{p}(c)$ будет существовать функция $p_{c,\alpha} : C \to [0; 1]$, для которой $p_{c,\alpha}(c) = \alpha$, $\forall x \in C$ выполнено $p_{c,\alpha}(x) \in \mathbf{p}(x))$ и $(C, p_{c,\alpha})$ — непротиворечивый [33].

Для дальнейшей работы с математическими моделями фрагментов знаний и их наборами удобно определить вес, который соответствует каждому ФЗ — *нагрузкой* или *весом* ФЗ $W(C, p)$ назовем подалфавит алфавита, над которым задан ФЗ: $W(C, p) = \{x_i \mid x_i \in C, x_i \in A\}$ [32].

Каждый ФЗ очень тесно характеризует связи между атомарными пропозициями, которые в него входят, и сложность их обработки, как, например, проверка непротиворечивости, растет экспоненциально с увеличением количества атомов. Поэтому, как предполагается в теории алгебраических байесовских сетей, все утверждения разбиваются на группы, фрагменты знаний, имеющие тесные связи между внутренними элементами. Таким образом можно рассматривать наборы фрагментов знаний и, соответственно, наборы моделей этих фрагментов знаний. При этом возможен случай, когда один вес одного ФЗ полностью включается в вес некоторого другого ФЗ и подобные ситуации необходимо исключить ввиду избыточности информации в противном случае. Поэтому назовем *набором максимальных моделей фрагментов знаний* (набор МФЗ, или просто МФЗ, первичная структура алгебраической байесовской сети) такой набор математических моделей фрагментов знаний, что никакая нагрузка ФЗ не содержится полностью в нагрузке другого ФЗ из представленного набора [32]. То есть $\forall i \neq j$ выполнено: $W(V_i) \not\subset W(V_j)$ и $W(V_j) \not\subset W(V_i)$.

Как и отдельные ФЗ, первичная структура также может быть противоречива, причем в таком случае определяется несколько степеней, в частности

- АБС *экстернально непротиворечива*, если каждый ФЗ в АБС непротиворечив и оценки вероятности истинности каждого конъюнкта, входящего в более чем один ФЗ, совпадают с оценками вероятности этого коннъюнкта в других ФЗ [8].
- АБС *интернально непротиворечива*, если каждый ФЗ в АБС непротиворечив, а также для любого конъюнкта из АБС и для любого скалярного значения, взятого из интервала оценки его истинности, есть способ выбрать совпадающие на одинаковых формулах скалярные значения во всех остальных ФЗ в АБС таким образом, что все ФЗ с точечными оценками будут непротиворечивы [8].
- АБС *глобально непротиворечива*, если ее, с ее оценками вероятностей, можно погрузить в объемлющий непротиворечивый ФЗ таким образом, что оценки на формулах АБС не поменяются [8].

После построения алгебраической байесовской сети начнется этап ее применения. На данной стадии возможно изменение изначальных параметров модели, и для этих случаев, в частности, рассматривается такой математический объект, как свидетельство. Он моделирует новую информацию о предметной области. Например, если имеется сформированная модель, описывающая прогноз погоды, то новой информацией будет утверждение о том,

что завтра определенно пойдет снег или то, что завтра, скорее всего, температура будет выше 0 ℃. На основе этой информации необходимо построить свидетельство и учесть его в сформированной модели. Обработка поступающих или оценка возможных для поступления свидетельств является апостериорным выводом. При апостериорном выводе могут рассматриваться три вида свидетельства, детерминированное, стохастическое и неточное [35]:

- *Детерминированным свидетельством* назовем предположение о том, что часть атомов получили конкретное означивание. Обозначается как $\langle c_i, c_j \rangle$, где c_i и c_j — конъюнкты, которые состоят из атомов, получивших соответствующие положительные и отрицательные означивания [35].
- *Стохастическое свидетельство* — предположение о том, что над C', подыдеале C, задается непротиворечивый ФЗ со скалярными оценками, определяющий вероятности истинности соответствующих конъюнктов [35].
- *Неточное свидетельство* — предположение о том, что над C', подыдеале C, задается непротиворечивый ФЗ с интервальными оценками, который определяет вероятности истинности соответствующих элементов подыдеала [35].

Данные свидетельстваа можно локально распространять (пропагировать) в моделях фрагментов знаний, что описано в [35]. Также пропагацию свидетельства можно проводить и глобально, на всей алгебраической байесовской сети [8].

2.2. Вторичная структура АБС

Помимо рассмотрения простых наборов ФЗ существует способ их представления в виде графовой структуры, дополняющей модели фрагментов знаний связями между ними. Для изучения такого способа сперва дадим понятие *сепаратора*, общего для двух ФЗ. То есть сепаратором двух МФЗ, V_i и V_j, назовем *подалфавит*, который является пересечением нагрузок этих ФЗ: $W(V_i, V_j) = W(V_i) \cap W(V_j), i \neq j$ [12]. При этом пара МФЗ называются *сочлененными*, если их сепаратор непуст [12].

Новой глобальной структурой будет являться в таком случае *граф максимальных моделей фрагментов знаний* — ненаправленный граф, вершинам которого сопоставлены МФЗ, вошедшие в АБС и ребра возможны только между сочлененными ФЗ. Как и для модели фрагмента знаний, для ребра этого графа также удобно дать понятие веса [32]. *Нагрузкой* $W(\{V_i, V_j\})$ *ребра* $\{V_i, V_j\} \in E(G)$ графа G назовем сепаратор его концов: $W(\{V_i, V_j\}) = W(V_i) \cap W(V_j)$ [12]. Определим и *нагрузку* $W(H)$ *подграфа* $H \subseteq G$ — наибольший по включения подалфавит, входящий в нагрузку всех вершин подграфа: $W(H) = \bigcap_{V \in H} W(V)$ [12].

Для графов, построенных по вышеописанному принципу, характерен определенного вида путь. А именно рассмотрим *магистральный путь* между сочлененными вершинами V_i и V_j — такой путь между этими вершинами, что нагрузка каждой вершины пути содержит сепаратор концов этого пути [32]. Далее граф будет *магистрально связен*, если между каждой из его сочлененных вершин существует магистральный путь [32]. *Граф смежности* — магистрально связный граф МФЗ [32]. *Дерево смежности* — граф смежности, представимый в виде дерева [32].

Если имеется ацикличная АБС, то из ее интернальной непротиворечивости следует глобальная [8]. Заметим, что с учетом этого утверждения экстернально непротиворечивая АБС со скалярными оценками, представимая в виде дерева смежности (ацикличная АБС), глобально непротиворечива, так как экстернальная непротиворечивость для такой АБС влечет интернальную непосредственно по определению. Для поддержания и проверки интерналь-

ной, глобальной непротиворечивости АБС, непротиворечивости ФЗ применяется решение задач линейного программирования (ЗЛП) [8, 27], при этом проверка интернальной непротиворечивости (а тем более экстернальной) имеет значительно меньшую вычислительную сложность, чем проверка глобальной, поэтому утверждение о глобальной непротиворечивости, следуемой из интернальной (экстернальной) полезно при практической реализации проверки непротиворечивости [8].

В результате, помимо первичной структуры АБС, можно дать определение *вторичной*. Такой структурой будет являться некоторый граф смежности АБС [8].

Рассмотрим частные случаи вторичных структур. Например, *минимальный граф смежности* (МГС) — граф смежности, число ребер которого минимально [32].

Так же существует понятие *максимального графа смежности* G_{max} — наибольшего по числу ребер графа смежности [32]. Для заданного множества вершин существует единственный максимальный граф смежности, то есть тот, в котором между вершинами существует ребро только тогда, когда они сочлененные [32].

Следуя [34], дополнительно предположим, что первичная структура *связна*, то есть связен максимальный граф смежности, построенный над этой структурой. В противном случае можно рассматривать наборы вершин из каждой компоненты связности как отдельные АБС.

2.3. Третичная структура АБС

Далее перейдем к основному объекту, рассматриваемому в работе. Перед его непосредственным определением необходимо также введение нескольких новых понятий. Так, *сужением* $G \downarrow U$ графа G на нагрузку U назовем граф, в который входят те и только те ребра и вершины исходного графа G, нагрузки которых равны или содержат U [32]. *Значимое сужение* — сужение на нагрузку, которая является сепаратором для некоторой пары МФЗ [32]. На сужение можно наложить дополнительные ограничения, тогда получим *сильное сужение* $G \Downarrow U$ — сужение $G \downarrow U$, из которого удалили все ребра нагрузки U [34]. После сильного сужения граф $G \downarrow U$ разбивается на компоненты связности, после сужения же $G \downarrow U$ граф остается связным [34].

Одним из основных объектов в новоопределяемой структуре будет *значимая нагрузка* U — непустой сепаратор некоторой пары ФЗ первичной структуры [12]. *Замкнутым же снизу множеством нагрузок* назовем объединение множества значимых нагрузок с множеством нагрузок вершин МФЗ [12]. *Замкнутое множество нагрузок* — объединение замкнутого снизу множество нагрузок с одноэлементным множеством, содержащим пустое множество [12].

При этом на множестве нагрузок существует частичный порядок, являющийся отношением включения. Таким образом, *родительским графом* (*третичной структурой* АБС) назовем диаграмму Хассе замкнутого множества нагрузок [12]. Диаграмму Хассе можно рассматривать как транзитивное сокращение, поэтому родительский граф единственный при заданной первичной структуре АБС [12, 36].

3. Апостериорный вывод

В этом разделе опишем алгоритм использования третичной структуры в апостериорном выводе. В итоге получим основную процедуру `PosterioriInfer`, при применении которой будет пропагироваться соответствующее свидетельство.

Общее описание алгоритма апостериорного вывода с применением третичной структуры рассматривается в работе [12]. В ней доказывается теорема о том, что алгоритм завершит работу и после завершения строит оценки вероятностей, которые в случае точечных оценок совпадают с результатом работы алгоритма пропагации свидетельства по минимальному графу смежности, при этом рассматриваются только ациклические алгебраические сети. Рассмотрим это общее описание, взятое из [12]:

1. Распространить свидетельство в ФЗ, входящие в сужение $G_{max} \downarrow u$ для некоторой нагрузки u, содержащей нагрузку свидетельства. Повторная пропагация в ФЗ не проводится [12].
2. Пометить u и всех ее потомков. Если существуют нагрузки со всеми помеченными детьми, пометить их [12].
3. Выбрать непомеченную нагрузку максимальной мощности v (по количеству атомарных пропозиций), сыном которой является некоторая помеченная нагрузка w. Если все нагрузки помечены и выбрать v не удается, то завершить работу [12].
4. В противном случае, если удалось выбрать v, сформировать свидетельство нагрузка которого совпадает с v, а оценки вероятности взяты из какого-либо ФЗ w. Перейти к шагу 1 [12].

В работе также описывается, что данный алгоритм выполним, если нагрузка изначального свидетельства содержится хотя бы в одном узле родительского графа, иначе применяют метод распространения множества детерминированных свидетельств, рассматриваемый в работе [8].

Этот алгоритм (псевдокод), а также вспомогательные алгоритмы, представлены в листингах 1 и 2.

Пояснения к алгоритмам:

- Алгоритм `InferInSubgraph`
  - `Mark` и `IsMarked` — помечающие нагрузку и проверяющие на наличие метки процедуры;
  - `Propagate` — процедура, локально распространяющая свидетельство.
- Алгоритм `GenerateEvidenceIfPossible`
  - `SeparatorsNarrowing` — словарь, сопоставляющий каждому сепаратору ФЗ, входящих в сужение на него;
  - `GenerateEvidence(u, kp)` — функция, формирующая свидетельство для конъюнктов из `u` и оценок из `kp`;
  - `Any(s)` — функция, возвращающая некоторый (любой) элемент из множества s;
  - Можно увеличить скорость работы алгоритма, если при установлении метки оповещать родительские вершины об этом, а при просмотре родительских вершин просматривать количество меток у детей. Но для этого могут понадобиться дополнительные ссылки на родителей.
- Алгоритм `PosterioriInfer`
  - `SortedWeights` — все нагрузки, включая ФЗ и сепараторы, отсортированные в порядке уменьшения количества атомарных пропозиций. В конце стоит пустая нагрузка — вершина родительского графа;
  - `Contains` — функция, проверяющая включение свидетельства во ФЗ. Работает за $O(1)$.

Листинг 1 Вспомогательные алгоритмы, пропагирующие свидетельство в подграфе (`InferInSubgraph`) и формирующие свидетельство (`GenerateEvidenceIfPossible`) в соответствии с алгоритмом апостериорного вывода

```
1: ▷ Реализует шаг 1 и часть шага 2
2: procedure InferInSubgraph(e ∈ Evidences, w ∈ Weights)
3:     Mark(w)                                  ▷ В процессе пропагации помечаем вершины
4:     if w ∈ KnowledgePatterns then
5:         Propagate(e, w)                      ▷ Распространяем свидетельство в ФЗ
6:         return                               ▷ Вершина с нагрузкой в виде ФЗ — лист, поэтому потомков у нее нет. Эта строка для ясности алгоритма, без нее следующий цикл все равно бы не запустился
7:     for all child ∈ w.Children do            ▷ Распространяем свидетельство вниз по родительскому графу — от родителей к потомкам
8:         if not IsMarked(child) then          ▷ Повторная пропагация не проводится
9:             InferInSubgraph(e, child)
10:
11: ▷ Формирует свидетельство в 4 шаге. Свидетельсто формируется только тогда, когда у непомеченой v есть и помеченный, и непомеченный потомок. В противном случае v не подходит и возвращается Null
12: function GenerateEvidenceIfPossible(v ∈ Weight, sn ∈ SeparatorsNarrowing)
13:     if not IsMarked(v) then                  ▷ Сразу проверяем v на непомеченность
14:         hasMarkedChild ← false
15:         hasNotMarkedChild ← false
16:         kpWithEvidence ← Null                ▷ kpWithEvidence будет хранить ФЗ для w из 4-го шага
17:         for all w ∈ v.Children do
18:             if IsMarked(w) then
19:                 if not hasMarkedChild then
20:                     hasMarkedChild ← true
21:                     kpWithEvidence ← Any(sn[w])   ▷ Выбираем некоторый (любой) ФЗ, входящий в сужение на w
22:             else
23:                 hasNotMarkedChild ← true
24:             if hasMarkedChild and hasNotMarkedChild then
25:                 return GenerateEvidence(v, kpWithEvidence)
26:         if hasMarkedChild then
27:             Mark(v)                          ▷ Будут помечены все сыновья, так как hasNotMarkedChild = false, иначе возвратилось бы свидетельство в строке 24
28:     return Null
```

Листинг 2 Алгоритм апостериорного вывода с применением третичной структуры `PosterioriInfer`

```
1: ▷ Процедура, при применении которой пропагируется свидетельство e
2: procedure PosterioriInfer(e ∈ Evidences, pg ∈ ParentGraphs)
3:     weightToPropagate ← Null                          ▷ weightToPropagate будет хранить нагрузку для следующей пропагации
4:     for all w ∈ Reversed(pg.SortedWeights) do         ▷ Ищем начальную нагрузку для пропагации — наименьшую по включению нагрузку, содержащую e
5:         if Contains(w, e) then
6:             weightToPropagate ← w
7:             break
8:     if weightToPropagate = Null then                  ▷ В алгоритме рассматривается только случай, когда свидетельство содержится хотя бы в одной нагрузке
9:         return
10:    while not IsMarked(pg.EmptyWeight) do             ▷ Пока не помечен весь граф (если помечена пустая нагрузка, то помечены все ее потомки, а значит и весь граф)
11:        InferInSubgraph(e, weightToPropagate)         ▷ Распространяем свидетельство в подграф с корнем в выбранной нагрузке
12:        for all v ∈ pg.SortedWeights do               ▷ Первая часть 3-го шага. Выбираем нагрузку v
13:            e ← GenerateEvidenceIfPossible(v, pg.SortedWeights)   ▷ Формируем новое свидетельство (в 4 шаге)
14:            if e ≠ Null then
15:                weightToPropagate ← v                 ▷ Выбираем v
16:                break
17:    if pg.HasIntervalKP or e.IsInterval then
18:        KeepInternalConsistency(pg)                   ▷ Если оценки получились интервальными, то поддерживаем интернальную непротиворечивость
```

Утверждение 1. Алгоритм `PosterioriInfer` в случае скалярных оценок назначает оценки вероятностей, совпадающие с результатом пропагации по МГС, а в случае интервальных оценок строит накрывающие оценки, которые бы соответствовали результату пропагации в МГС, если бы распространение виртуального свидетельства давало точные оценки.

Доказательство. Для начала рассмотрим соответствие представленного алгоритма и общего описания. В целом, происходит непосредственная реализация общего описания, с тем лишь отличием, что установление метки на вершины, для которых все сыновья помечены, происходит во время выбора вершины u в шаге 3. Алгоритм отработает корректно, так как сыновья включают большее число атомарных пропозиций, поэтому они, рассматривая `sortedWeights`, будут правильно помечены перед рассмотрением родительской вершины.

Случай скалярных оценок разобран в работе [12], показывающий, что при данных условиях алгоритм строит оценки вероятностей, которые совпадают с результатом пропагации свидетельства по минимальному графу смежности.

Разберем работу алгоритма с интервальными оценками вероятности истинности. В таком случае пропагация виртуального свидетельства между двумя ФЗ предполагает нахождение экстремальных значений, сопоставляемых итоговым оценкам, получаемым после пропагации. При этом при нахождении соответствующих минимумов и максимумов рассматриваются все элементы из семейств распределений, соответствующих оценкам как свидетельства, так и самого ФЗ [37]. Поэтому, если свидетельство и/или ФЗ имели накрывающие интервальные оценки, то оценки вероятности истинности, полученные в таком случае после распространения свидетельства, будут накрывающими по отношению к оценкам, полученным в результате пропагации с точными оценками.

Далее, как показывается в работе [12], после шага 2 подграф G' минимального графа смежности G, состоящий из помеченных ФЗ, остается связен, а подграф, получаемый из непомеченных вершин, разбивается на несколько компонент связности, и пропагация в МГС в каждую компоненту связности c_i в происходит по единственному ребру, соединяющему G' и c_i. В таком случае пропагация свидетельства на новые компоненты связности c_i в МГС будет соответствовать пропагации с использованием алгоритма с тем лишь отличием, что могут использоваться разные накрывающие оценки для виртуальных свидетельств и получаться таким образом различные накрывающие апостериорные оценки.

Так же стоит добавить, что в результате пропагации по алгоритму могут получится интернально противоречивые накрывающие оценки, поэтому необходимо обеспечивать соответствующую непротиворечивость, которая, в случае ацикличной первичной структуры даст глобальную. □

Утверждение 2. Сложность работы алгоритма апостериорного вывода с применением третичной структуры в случае скалярных оценок лежит в классе $O(w(w \cdot O(\texttt{Propagate}) + s_{all}e_s + c))$, где w — общее количество ФЗ, $O(\texttt{Propagate})$ — сложность функции, локально распространяющей свидетельство, s_{all} — количество вершин в родительском графе, e_s — количество ребер в графе смежности, c — максимальное число конъюнктов в ФЗ.

Доказательство. Рассмотрим сложность `InferInSubgraph`. Алгоритм в строке 7 рассматривает все ребра в подграфе родительского графа, содержащие нагрузку u, в количестве e_u. Дойдя до листьев-ФЗ, рассматриваемых в количестве w_u штук, алгоритм пропагирует в них свидетельство (строка 5). Таким образом сложность алгоритма будет выражаться как $O(w_u \cdot O(\texttt{Propagate}) + e_u)$. При этом $w_u \leq w$ и $e_u \leq e_s$, отсюда следует более общий класс, в котором лежит сложность `InferInSubgraph`, равный $O(w \cdot O(\texttt{Propagate}) + e_s)$.

Далее оценим сложность алгоритма `GenerateEvidenceIfPossible`. Алгоритм рассматривает всех детей вершины u в количестве ch_u (строка 17) и, возможно, генерирует свидетельство (строка 25), беря оценки из ФЗ KPWithEvidence. Взятие оценок потребует не больше чем c операций, и $ch_u \leq e_s$. Поэтому сложность алгоритма лежит в классе $O(e_s + c)$.

Перейдем к алгоритму `PosterioriInfer`. В строке 4 рассматриваются не более чем s_{all} элементов, где s_{all} — количество вершин в родительском графе. Далее заметим, что на каждом шаге цикла while происходит пропагация свидетельства хотя бы в один новый ФЗ, поэтому количество итераций этого цикла не будет превосходить w. На каждом шаге происходит работа алгоритма `InferInSubgraph`, затем выполняется не более чем s_{all} раз работа `GenerateEvidenceIfPossible`. Замечу, что создание нового свитедельства в `GenerateEvidenceIfPossible` происходит один раз за итерацию цикла while, поэтому оцен-

ку сложности всего алгоритма можно представить в виде $O(s_{all} + w(w \cdot O(\texttt{Propagate}) + e_s + s_{all}e_s + c))$, что лежит в классе $O(w(w \cdot O(\texttt{Propagate}) + s_{all}e_s + c))$. □

В случае интервальных оценок добавляется сложность поддержания интернальной непротиворечивости.

4. Проверка ацикличности

В этом разделе решим задачу проверки ацикличности алгебраической байесовской сети с использованием только третичной структуры. В результате опишем функцию `GetComponentQuantity`, после применения которой получим общее число компонент связности, что в сочетании со следующей теоремой, описанной в [31], поможет дать ответ на вопрос об ацикличности алгебраической байесовской сети:

Теорема 1. Связная первичная структура АБС циклична тогда и только тогда, когда не выполняется соотношение:

$$|\text{МКР}| = \sum_{U \in Sep} Conn(G_{max} \downarrow U) - |Sep| + 1,$$

где МКР — первичная структура АБС, набор ФЗ, $Conn(G_{max} \downarrow U)$ — число компонент связности графа $G_{max} \downarrow U$, Sep — множество непустых сепараторов.

Для того, чтобы применить эту теорему на практике, необходимо рассчитать каждое из слагаемых выражения. Мощность первичной структуры равна количеству листьев-ФЗ в родительском графе, что можно посчитать при построении родительского графа. Количество вершин родительского графа будет соответствовать количеству сепараторов, исключая пустую, верхнюю вершину и листья. Таким образом, остается получить число компонент связности с использованием третичной структуры. Для этого, сперва, докажем теорему:

Теорема 2. Возьмем пару вершин с нагрузками в виде ФЗ kp_1 и kp_2 и сильное сужение на значимую нагрузку u. Тогда эти вершины лежат в одной компоненте связности C_u графа $G_{max} \downarrow u$ в том и только том случае, если существует такая последовательность вершин с нагрузками в виде ФЗ, что вершины с нагрузками kp_1 и kp_2 являются крайними, а также для любой пары соседних вершин v_i и v_{i+1} в этой последовательности существует нагрузка, которая является одновременно и предком по отношению к v_i и v_{i+1}, и потомком по отношению к вершине с нагрузкой u в родительском графе. Формально:

$$kp_1, kp_2 \in C_u \Leftrightarrow \exists v_1 = kp_1, v_2, ..., v_{n-1}, v_n = kp_2 :$$

$$\forall i = 1, ..., n-1 : (\exists w : v_i, v_{i+1} \in \text{descendants}(w) \;\&\; w \in \text{descendants}(u)).$$

Доказательство. Пусть нашлась последовательность как в условии теоремы, связывающая kp_1 и kp_2 в графе $G_{max} \downarrow u$. Рассмотрим пару соседних элементов последовательности. Так как между этими элементами существует вершина w, которая является предком по отношению к ним, то ребро, соединяющее эти элементы, включает вес w. С другой стороны, w — потомок по отношению к u, поэтому нагрузка связующего ребра будет содержать, но не равняться u. Таким образом, существует путь в $G_{max} \downarrow u$ между вершинами с нагрузками kp_1 и kp_2, поэтому они лежат в одной компоненте связности C_u.

Пусть kp_1 и kp_2 лежат в одной компоненте связности C_u. Тогда между ними существует путь, который связан ребрами, нагрузки которых содержат, но не равняются нагрузке сильного сужения u. Следовательно, существует указанная в условии теоремы последовательность вершин. □

Замечание 1. Рассмотрим две вершины w_1 и w_2 в родительском графе. Если существует вершина v, являющаяся для них общим потомком, то все ФЗ, которые являются потомками по отношению к w_1 и w_2, лежат в одной компоненте связности, которая была получена в результате сильного сужения на новую, общую для вершин w_1 и w_2 вершину-предка u. Более формально: если $\exists w_1, w_2 : kp_1 \in \text{descendants}(w_1)$ & $kp_2 \in \text{descendants}(w_2)$ & $w_1, w_2 \in \text{descendants}(u)$, а также $\exists v : v \in \text{descendants}(w_1) \cap \text{descendants}(w_2)$, то $kp_1, kp_2 \in C_u$.

Таким образом, предоставим алгоритм подсчета числа компонент связности, опирающийся на доказанную теорему. Предположим, что нужно найти число компонент связности графа $G_{max} \downarrow u$. Для этого будем распространять по родительскому графу от вершины u маркеры, назовем их *цветами*. Сначала распространим от u *различные* цвета до каждой вершины-потомка u. Далее, от каждой вершины-потомка u распространим по уже их вершинам-потомкам цвет, который был получен ранее. В том случае, если в некоторую вершину поступало несколько отличающихся цветов, то тогда примем эти цвета одинаковыми. В конечном счете рассмотрим количество различных цветов, которые остались после распространения цветов до листьев родительского графа. Это количество и будет соответствовать количеству компонент связности $G_{max} \downarrow u$. Пример распространения цветов показан на рис. 1.

Покажем, что алгоритм корректен. Предположим, что нашлись две вершины с нагрузками в виде ФЗ, которые получили одинаковый цвет. Тогда, исходя из замечания 1, эти вершины будут лежать в одной компоненте связности $G_{max} \downarrow u$. Далее допустим, что некоторые две вершины родительского графа с нагрузками kp_1 и kp_2 получили различные цвета, но тем не менее лежат в единой компоненте связности. В этом случае, в соответствии с теоремой 2, должна существовать последовательность из вершин $v_1, ..., v_n$, где каждая пара соседних вершин имеет по построению одинаковый цвет. Но тогда $v_1 = kp_1$ и $v_n = kp_n$ будут окрашены в один цвет — противоречие.

При реализации рассмотренного алгоритма возникает вопрос о том, как эффективно хранить и объединять цвета. В работе предлагается распространять вместо цветов числа вида $2^i, i = 0, ..., n-1$, где n — число дочерних узлов начальной вершины, рассматриваемой для распространения. В таком случае объединение цвета — применение побитового «или».

Далее необходимо понять то, как будут распространяться цвета. Предположим, что мы будем распространять от каждой вершины все цвета, которые могли дойти до этой вершины. В противном случае было бы необходимо для каждой вершины хранить отдельно множества цветов и каким-то образом объединять схожие цвета, перебирать эти множества на каждом шаге распространения. Такой алгоритм интуитивно более ресурсозатратен и его изучение не вошло в рамки данной статьи.

Итак, если использовать для распространения цвета, например, обход в ширину, то может произойти ситуация, при которой передача цветов произойдет до того, как вершина получит информацию о всех приходящих цветах, как в примере на рис. 2. В данной работе предлагается при построении родительского графа хранить в вершинах число родителей и

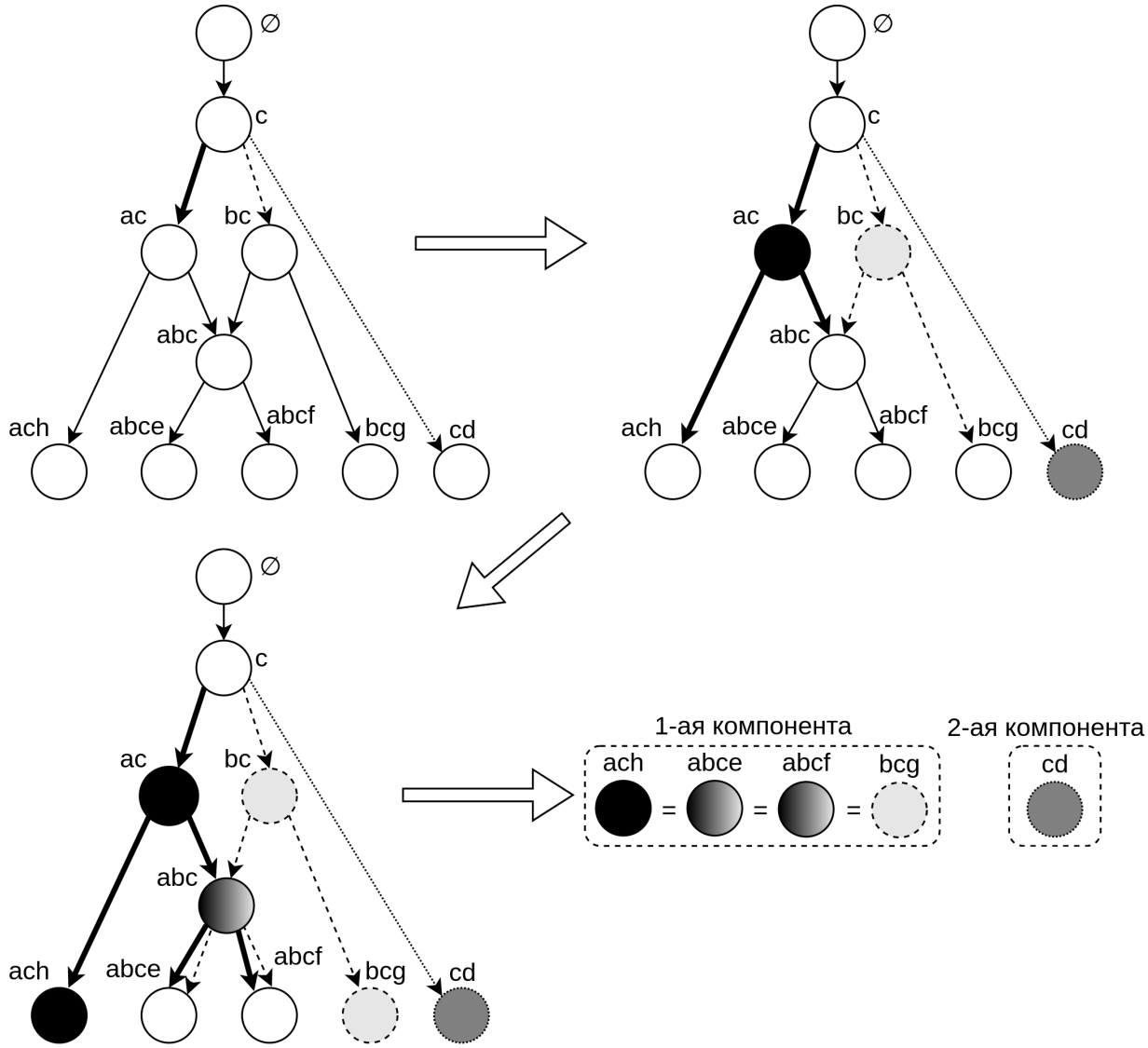

Рис. 1. Распространение цветов при сужении на вершину c. Всего три цвета, вершины помечаются различными цветами, ребра — толщиной ребра и пунктиром с отличающимся интервалом между штрихами

проводить распространение цвета только после того, как вершину просмотрели все родительские узлы. Алгоритм распространения цветов `SpreadColors` представлен в листинге 3.

Пояснения к `SpreadColors`:

- `WeightsToColors` — словарь, сопоставляющий цвет сепаратору. Относительно каждого сепаратора-предка вершина будет иметь свой цвет, который будет передаваться дочерним вершинам. Его можно удалять у вершины, если полностью произошло распространение цветов к дочерним узлам;
- `NumberOfParents` задается при построении родительского графа, `ParentCount` так же можно инициализировать нулем при построении графа;
- & и | — побитовые операции «и» и «или» соответственно.

Утверждение 3. В результате работы процедуры `SpreadColors` каждый ФЗ получит корректное распределение цветов.

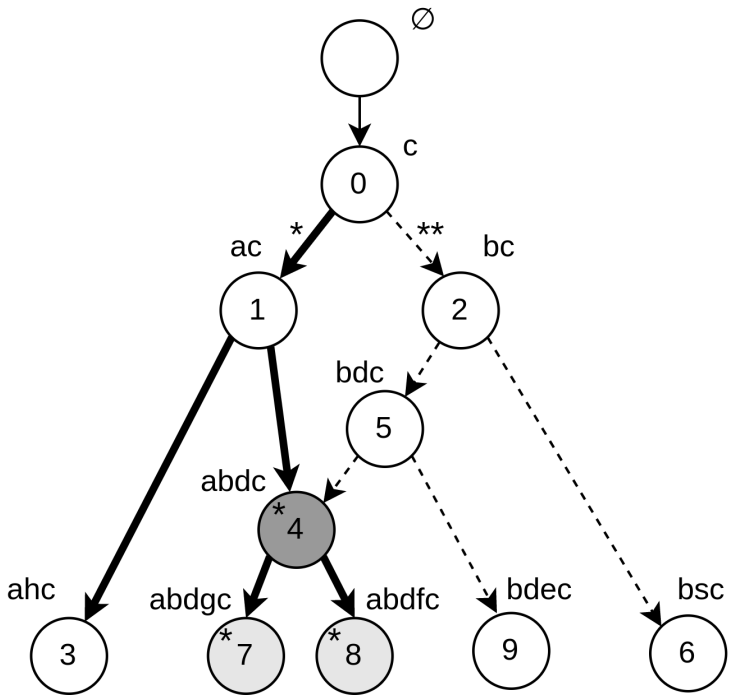

Рис. 2. Возникновение ошибки распространения цвета (4-я вершина) при обходе в ширину. Всего два цвета, они помечены как «*» и «**». Вершина №4 должна распространить оба цвета, но продвигается только «*». Номера вершин соответствуют порядку обхода

Доказательство. Действительно, реализуется непосредственно вышеописанный алгоритм, при этом цвет распространяется от вершины только тогда, когда получены цвета от всех родительских вершин. □

Утверждение 4. Сложность работы процедуры `SpreadColors` лежит в классе $O(e_s s)$, где e_s — количество ребер в родительском графе, а s — количество непустых сепараторов.

Доказательство. В строке 4 проверяются все дочерние узлы текущей вершины, при этом каждая вершина рассматривается как родительская только один раз. Соответственно, таких проверок будет e_s штук. При этом в 5 строке при каждой проверке перебираются все все сепараторы из `WeightsToColors`, которых может быть не более s. Все остальные шаги алгоритма имеют сложность класса $O(1)$, из чего следует искомая сложность алгоритма. □

Конечным этапом будет последнее объединение цветов, дошедших до листьев-ФЗ. Его идея заключается в том, что сначала формируется список наборов цветов, дошедших до ФЗ и соответствующих каждому сепаратору. Затем каждый представляемый в виде числа набор цветов, продвигаясь по списку, сравнивается с остальными наборами и объединяется в том случае, если есть хотя бы одно пересечение, при этом использованные наборы цветов маркируются. Далее последний объединяемый набор помечается особым образом и на его место в списке записывается совокупность цветов, полученных при текущей итерации объединений. Записываемые после распространения цвета, *последние цвета*, могут далее участвовать в объединении, в отличие от промаркированных простым образом. В итоге, после работы такого алгоритма количество последних цветов и будет соответствовать ко-

Листинг 3 Алгоритм распространения цветов до листьев-ФЗ `SpreadColors`

```
1: procedure SpreadColors(w ∈ Weights)
2:     if w ∈ KnowledgePatterns then
3:         return                                  ▷ Вершина с нагрузкой в виде ФЗ — лист, поэтому
                                                     потомков у нее нет, распространять цвета даль-
                                                     ше не нужно. Эта строка для ясности алгорит-
                                                     ма, без нее следующий цикл все равно бы не
                                                     запустился
4:     for all v_i ∈ w.Children do                 ▷ Перебираем дочерние вершины для распростра-
                                                     нения цветов
5:         for all s ∈ w.WeightsToColors do        ▷ Рассматриваем все сепараторы, относительно
                                                     которых нагрузка w получила цвета
6:             if s ∈ v_i.WeightsToColors then     ▷ Если v_i ранее получала цвет относительно сепа-
                                                     ратора s
7:                 ▷ То добавляем цвет от w
8:                 v_i.WeightsToColors[s] ← v_i.WeightsToColors[s] | w.WeightsToColors[s]
9:             else
10:                ▷ В противном случае кладем к v_i новый цвет
11:                v_i.WeightsToColors[s] ← w.WeightsToColors[s]
12:        v_i.WeightsToColors[w] ← 2^i            ▷ В каждую v_i кладем свой цвет относительно w
13:        v_i.ParentCount ← v_i.ParentCount + 1   ▷ Учитываем, что для одного из родителей v_i про-
                                                     изошла передача цвета
14:        if v_i.NumberOfParents = v_i.ParentCount then
15:            SpreadColors(v_i)                   ▷ Продолжаем распространять цвета от дочерних
                                                     вершин только в том случае, когда были полу-
                                                     чены цвета от всех родителей
```

личеству компонент связности. Этот алгоритм (`GetComponentQuantity`) описан в листинге 4, пример его работы изображен на рис. 3.

Пояснения к `GetComponentQuantity`:

- `GetListOfColors` — функция, получающая список итоговых наборов цветов, дошедших до ФЗ;
- `SimplyMark(i)` и `MarkAsLast(i)` — взаимозаменяющие процедуры, помечающие элемент с индексом i самого списка, `IsSimplyMarked` и `IsMarkedAsLast` — процедуры, проверяющие то, как помечены элементы. Могут быть реализованы через создание массива, по количеству элементов равному количеству ФЗ, которые рассматриваются для текущего сепаратора, и хранящего соответствующие маркировки.
- `QuantityOfMarkedAsLast` — функция, возвращающая количество помеченных как последние элементов. Может реализовываться, например, через перебор массива.

Утверждение 5. В результате работы функции `GetComponentQuantity` возвратится общее количество компонент связности.

Доказательство. Начнем с того, что процедура `SpreadColors` корректно распределит цвета по листьям-ФЗ. Далее заметим, что каждый из начальных наборов цветов списка (где набор цветов — сумма соответствующих чисел 2^i, дошедших до ФЗ) будет полностью лежать

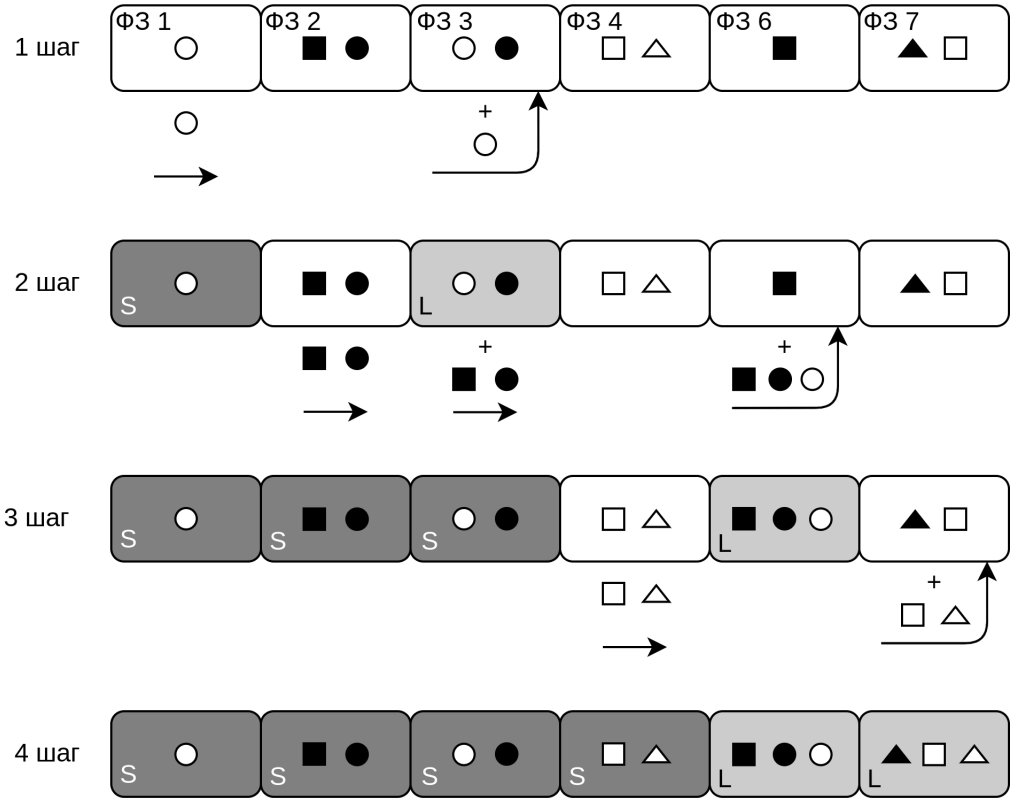

Рис. 3. Часть работы алгоритма `GetComponentQuantity` с вычислением последних объединений цветов для одного из сепараторов, где в итоге получается 2 компоненты связности. Цвета из алгоритма обозначены фигурами различной формы и цвета. Темно-серый цвет с пометкой «S» — «простая» маркировка, светло-серый цвет с пометкой «L» — «последние» элементы

в некотором элементе, помеченном как последний. Это основывается на том, что каждая вершина, соответствующая `IsSimplyMarked`, могла быть помечена только при объединении с каким-либо распространяющимся набором цветов, положенным в итоге в некоторую последнюю вершину.

Назовем путем наборов цветов такую последовательность изначальных наборов цветов, что каждая пара соседних элементов имеет одинаковый цвет. Заметим также, что если два первоначальных цвета лежат в последнем элементе, то, по построению, существует путь наборов цветов, который соединяет эти два цвета (то есть первый цвет лежит в первом наборе, а последний — в последнем).

С другой стороны, предположим, что существует путь, соединяющий некоторые два цвета. Каждая пара соседних наборов в этом пути должна лежать в одном и только одном последнем элементе, ведь при распространении общего для них цвета объединяющий набор цветов ($color_i$ в алгоритме) соединит эти наборы (даже если один уже был в некоторой последнем элементе) и добавит их в новый последний элемент. Объединение произойдет, так как только один из них не может быть промаркирован простым способом. При этом, каждый набор цветов попадет только в один последний элемент, так как он после объединения маркируется и не учитывается более. Итак, каждая пара наборов цветов будет лежать в одном последнем элементе, поэтому и весь путь будет лежать в нем.

Листинг 4 Алгоритм вычисления количества компонент связности для всех сильных сужений на сепараторы `GetComponentQuantity`

```
1: function GetComponentQuantity(pg ∈ ParentGraphs)
2:     SpreadColors(pg.MainNode)                          ▷ Распространяем цвета перед тем, как посчитать различные цвета в листьях-ФЗ
3:     totalComponentQuantity ← 0                         ▷ Будет хранить общее число компонент связности
4:     for all s ∈ Separators do                          ▷ Перебираем все сепараторы для определения количества различных цветов
5:         colors ← GetListOfColors(s, pg.KnowledgePatterns)   ▷ Получаем цвета, дошедшие до листьев-ФЗ
6:         for all color_i ∈ colors do                    ▷ Будем рассматривать цвета, идя вперед по массиву цветов и объединяя одинаковые
7:             if not IsSimplyMarked(i) and
                  not IsMarkedAsLast(i) then              ▷ Рассматриваем только непромаркированные элементы, так как для маркированных элементов распространение цвета вперед по массиву цветов уже произошло
8:                 SimplyMark(i)                          ▷ Помечаем как рассмотренный (*простым* образом)
9:                 lastHandled ← i                        ▷ Хранит индекс последнего элемента, с которым произошло объединение
10:                for all color_j ∈ colors[j > i] do
11:                    if color_i & color_j ≠ 0 and
                          not IsSimplyMarked(j) then      ▷ Если цвета одинаковые и если элемент не промаркирован *простым* образом (если элемент промаркирован простым образом, то нет смысла его рассматривать, так как цвет, который он содержит, будет содержаться в некотором *последнем* элементе, идущем далее), то
12:                        color_i ← color_i | color_j    ▷ Объединяем цвета. Меняется только переменная color_i, но не элемент массива
13:                        SimplyMark(j)
14:                        lastHandled ← j
15:                MarkAsLast(lastHandled)                ▷ Отдельно помечаем *последний* элемент. Их количество будет соответствовать количеству компонент связности
16:                colors[lastHandled] ← color_i          ▷ Записываем объединенный цвет, далее он также может сравниваться и объединяться
17:        totalComponentQuantity ← totalComponentQuantity +
               QuantityOfMarkedAsLast(colors)             ▷ Увеличиваем счетчик общего числа компонент связности, рассчитав количество компонент для рассмотренных на данном шаге цветов
18:    return totalComponentQuantity
```

Суммируя, получаем, что если два цвета должны быть связаны, то они не будут лежать в разных последних элементах, а будут лежать в одном, и при этом каждые два цвета в последнем элементе будут связаны, из чего следует доказательство утверждения. □

Утверждение 6. Сложность работы функции `GetComponentQuantity` лежит в классе $O(sw^2 + e_s s)$, где w — общее количество ФЗ.

Доказательство. Сначала заметим, что сложность работы `SpreadColors` лежит в классе $O(e_s s)$, по утверждению 4. Далее, в строке 4 перебираются все сепараторы, количество которых — s. Для каждого сепаратора в строке 6 рассматриваются все ФЗ, до которых дошел цвет этого сепаратора. Затем для этого цвета вновь просматриваются наборы цветов, которых не более чем w. Сложность работы функций `GetListOfColors` и `QuantityOfMarkedAsLast` лежит, как максимум, в классе $O(w)$. Все остальные шаги имеют сложность $O(1)$. В итоге, суммируя, получаем искомую сложность. □

Стоит отметить, что при небольших модификациях и дополнениях к данному алгоритму можно получить информацию о сужениях и сильных сужениях на все значимые нагрузки, которая представлена в соответствующих наборах цветов, принадлежащих ФЗ.

5. Результаты и обсуждение

В данной работе были представлены алгоритмы, позволяющие применять только третичную структуру АБС в глобальном апостериорном выводе. Так непосредственно был расширен до возможности применения в случае интервальных оценок существующий до написания работы алгоритм [12], способный ранее производить глобальный апостериорный вывод только в случае скалярных оценок. Это было достигнуто за счет поддержания интернальной непротиворечивости. Также алгоритм описан более подробно, то есть представлен в виде псевдокода. Таким образом, создана процедура `PosterioriInfer`, при применении которой будет пропагироваться соответствующее свидетельство. Процедура `PosterioriInfer` использует вспомогательную функцию `GenerateEvidenceIfPossible`, формирующее свидетельство, а также процедуру `InferInSubgraph`, распространяющую свидетельство в подграф родительского графа. Доказана корректность работы расширенного алгоритма, а именно показано, что процедура `PosterioriInfer` в случае скалярных оценок назначает оценки вероятностей, совпадающие с результатом пропагации по МГС, а в случае интервальных оценок строит накрывающие оценки, которые бы соответствовали результату пропагации в МГС, если бы распространение виртуального свидетельства давало точные оценки. Доказана сложность работы процедуры `PosterioriInfer`, которая в случае применения скалярных оценок лежит в классе $O(w(w \cdot O(\texttt{Propagate}) + s_{all}e_s + c))$, где w — общее количество ФЗ, $O(\texttt{Propagate})$ — сложность функции, локально распространяющей свидетельство, s_{all} — количество вершин в родительском графе, e_s — количество ребер в графе смежности, c — максимальное число конъюнктов в ФЗ. В случае интервальных оценок добавляется сложность поддержания интернальной непротиворечивости.

Допущением этого алгоритма является то, что его корректная работа определяется для ацикличных первичных структур. Поэтому был также представлен алгоритм, позволяющий проверять, представима ли вторичная структура в виде дерева смежности. Алгоритм проверки ацикличности использует только третичную структуру. Он основан на ранее доказанной теореме 1, критерии цикличности, который связывает количество моделей фрагментов знаний в сети с количеством непустых сепараторов и количеством компо-

нент связности сильных сужений в цикличной АБС. Для проверки критерия (равенства) необходимо рассчитать используемые в нем слагаемые. Основную сложность представляет расчет числа компонент связности сильных сужений, для этого была описана функция `GetComponentQuantity`. Работа этой функции опирается на доказанную в статье теорему 2 о принадлежности двух моделей фрагментов знаний к одной компоненте связности сильного сужения. Благодаря этой теореме в родительском графе можно производить определенное в статье распространение цветов от вершин графа к листьям. Распространение цветов вынесено в отдельную процедуру `SpreadColors`, которая используется в `GetComponentQuantity`. После распространения необходимо подсчитать количество различных цветов, где каждый цвет после распространения будет соответствовать одной компоненте связности, при этом цвета могут стать одинаковыми в процессе распространения. Подсчет одинаковых цветов занимает оставшуюся часть функции `GetComponentQuantity`. Таким образом рассчитывается число компонент связности всех сильных сужений, а вместе с тем и условие критерия. Доказана корректность работы этого алгоритма, а также найдена оценка сложности `GetComponentQuantity`, которая лежит в классе $O(sw^2 + e_s s)$, где s — количество непустых сепараторов, w — общее количество ФЗ, а e_s — количество ребер в родительском графе. Стоит отметить, что в результате работы алгоритма находится информация о всех сужениях и сильных сужениях максимального графа смежности на значимые нагрузки, что может использоваться в других алгоритмах, в частности, в описанном в текущей статье алгоритме апостериорного вывода.

Алгоритмы, представленные в работе, были автоматизированы и добавлены в веб-приложение [38]. Данное приложение содержит реализацию не только описанных в статье алгоритмов, но и других связанных с алгебраическими байесовскими сетями. По результатам проведенных исследований была добавлена реализация описанного в статье алгоритма глобального апостериорного вывода, использующего третичную структуру [39], а также алгоритма по проверке ацикличности [40].

В дальнейшем предполагается анализ времени работы данных алгоритмов и сравнение его со временем работы других алгоритмов, решающих те же задачи. Так, алгоритм проверки ацикличности планируется сравнить с алгоритмом, использующим еще одну, новую структуру (четвертичную) [30]. Апостериорный же вывод может проводиться с применением вторичной структуры — механизм распространения виртуальных свидетельств [8]. Основную сложность здесь представляет само построение вторичной структуры [29]. Оценки сложности этих алгоритмов используют понятия, выходящие за рамки данной статьи, но предполагается, что оценки алгоритмов проверки ацикличности сопоставимы по времени, а алгоритм апостериорного вывода, использующий механизм виртуального распространения свидетельств, будет уступать алгоритму, представленному в статье, за счет увеличения времени построения вторичных структур. Сравнение предполагается производить над различными ациклическими АБС. В частности, стоит рассмотреть время работы алгоритмов при различном количестве ФЗ, так как, например, построение вторичных структур будет требовать все больше времени при увеличении количества ФЗ.

Следует добавить, что автоматизация подобных алгоритмов может быть использована в практическом применении алгебраических байесовских сетей, например, для исследования социоинженерных атак [5, 6].

Заключение

Данная статья направлена на решение задачи по расширению и анализу алгоритма глобального апостериорного вывода в АБС, который применяет только третичную структуру [12]. Расширение заключается в дополнении алгоритма до возможности его применения в случае интервальных оценок, ранее использовавшегося только со скалярными оценками. Оно достигнуто за счет поддержания интернальной непротиворечивости. Алгоритм, рассматриваемый в [12] также описан более подробно, то есть представлен в виде псевдокода (процедура `PosterioriInfer`). Доказана корректность работы расширенного алгоритма, а именно показано, что процедура `PosterioriInfer` в случае скалярных оценок назначает оценки вероятностей, совпадающие с результатом пропагации по МГС, а в случае интервальных оценок строит накрывающие оценки, которые бы соответствовали результату пропагации в МГС, если бы распространение виртуального свидетельства давало точные оценки. Помимо этого доказана сложность работы процедуры `PosterioriInfer`.

При этом важным свойством АБС в таком случае является ацикличность, так как только для ацикличных АБС доказана корректность вышеупомянутого алгоритма апостериорного вывода. Таким образом, также изучался вопрос о проверке ацикличности АБС с применением только третичной структуры, подобные алгоритмы ранее не описывались. Предъявленный алгоритм опирается на ранее доказанный критерий ацикличности, представленный в виде равенства (теорема 1). В результате была описана функция `GetComponentQuantity`, которая рассчитывает число компонент связности сильных сужений — одно из слагаемых равенства теоремы, расчет которого представляет наибольшую сложность. Доказана корректность работы этого алгоритма, а также найдена оценка его сложности.

Теоретическая значимость работы заключается в возможном использовании ее результатов в изучении задачи апостериорного вывода в алгебраических байесовских сетях, исследовании третичной структуры как объекта, единственно достаточного для практического применения алгебраических байесовских сетей. Практическая значимость использования результатов заключается в потенциальном ускорении работы алгоритмов апостериорного вывода за счет уменьшения количества формируемых объектов.

В дальнейшем планируется анализ времени работы данных алгоритмов и сравнение его со временем работы других схожих по решаемым задачам алгоритмов, которые направлены на проверку ацикличности [30], а также на глобальный апостериорный вывод, но с использованием вторичных структур [8, 29].

Работа выполнена в рамках проекта по государственному заданию СПБ ФИЦ РАН СПИИРАН № FFZF-2022-0003.

Литература

1. Larrañaga P., Moral S. Probabilistic graphical models in artificial intelligence // Applied Soft Computing. 2011. Vol. 11, no. 2. P. 1511–1528. DOI: 10.1016/j.asoc.2008.01.003.
2. Yang Y., Xu M., Wu W., *et al.* 3D Multiview Basketball Players Detection and Localization Based on Probabilistic Occupancy // 2018 Digital Image Computing: Techniques and Applications (DICTA). IEEE, 2018. P. 1–8. DOI: 10.1109/DICTA.2018.8615798.
3. Masmoudi K., Abid L., Masmoudi A. Credit risk modeling using Bayesian network with a latent variable // Expert Systems with Applications. 2019. Vol. 127. P. 157–166. DOI: 10.1016/j.eswa.2019.03.014.

4. Qiao W., Liu Y., Ma X., Liu Y. Human Factors Analysis for Maritime Accidents Based on a Dynamic Fuzzy Bayesian Network // Risk analysis. 2020. Vol. 40, no. 5. P. 957–980. DOI: 10.1111/risa.13444.

5. Khlobystova A.O., Abramov M.V., Tulupyev A.L. An Approach to Estimating of Criticality of Social Engineering Attacks Traces // International Conference on Information Technologies, Saratov, February 7–8, 2019. Vol. 199. Springer, 2019. P. 446–456. DOI: 10.1007/978-3-030-12072-6_36.

6. Корепанова А.А., Абрамов М.В., Тулупьева Т.В. Идентификация аккаунтов пользователей в социальных сетях «ВКонтакте» и «Одноклассники» // Семнадцатая Национальная конференция по искусственному интеллекту с международным участием. КИИ-2019: сборник научных трудов, Ульяновск, 21–25 октября, 2019. Т. 2. 2019. С. 153–163.

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

8. Тулупьев А.Л. Алгебраические байесовские сети: глобальный логико-вероятностный вывод в деревьях смежности. СПб.: Издательство «Анатолия», 2007. 40 с. Элементы мягких вычислений.

9. Фильченков А.А. Алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик-собственников // Информатика и автоматизация. 2010. № 14. С. 150–169. DOI: 10.15622/sp.14.9.

10. Фильченков А.А. Алгоритм построения множества минимальных графов смежности при помощи клик-собственников владений // Информатика и автоматизация. 2010. № 15. С. 193–212. DOI: 10.15622/sp.15.10.

11. Фильченков А.А., Тулупьев А.Л. Третичная структура алгебраическое байесовской сети // Информатика и автоматизация. 2011. № 18. С. 164–187. DOI: 10.15622/sp.18.7.

12. Фроленков К.В., Фильченков А.А., Тулупьев А.Л. Апостериорный вывод в третичной полиструктуре алгебраической байесовской сети // Информатика и автоматизация. 2012. № 23. С. 343–356. DOI: 10.15622/sp.23.17.

13. Kabir S., Papadopoulos Y. Applications of Bayesian networks and Petri nets in safety, reliability, and risk assessments: A review // Safety Science. 2019. Vol. 115. P. 154–175. DOI: 10.1016/j.ssci.2019.02.009.

14. Amin M.T., Khan F., Ahmed S., Imtiaz S. A data-driven Bayesian network learning method for process fault diagnosis // Process Safety and Environmental Protection. 2021. Vol. 150. P. 110–122. DOI: 10.1016/j.psep.2021.04.004.

15. Baksh A.-A., Abbassi R., Garaniya V., Khan F. Marine transportation risk assessment using Bayesian Network: Application to Arctic waters // Ocean Engineering. 2018. Vol. 159. P. 422–436. DOI: 10.1016/j.oceaneng.2018.04.024.

16. Cai B., Kong X., Liu Y., *et al.* Application of Bayesian Networks in Reliability Evaluation // IEEE Transactions on Industrial Informatics. 2019. Vol. 15, no. 4. P. 2146–2157. DOI: 10.1109/TII.2018.2858281.

17. Wang Z., Chen C. Fuzzy comprehensive Bayesian network-based safety risk assessment for metro construction projects // Tunnelling and Underground Space Technology. 2017. Vol. 70. P. 330–342. DOI: 10.1016/j.tust.2017.09.012.

18. Tavana M., Abtahi A.-R., Caprio D.D., Poortarigh M. An Artificial Neural Network and Bayesian Network model for liquidity risk assessment in banking // Neurocomputing. 2018. Vol. 275. P. 2525–2554. DOI: 10.1016/j.neucom.2017.11.034.

19. Chaturvedi I., Ragusa E., Gastaldo P., *et al.* Bayesian network based extreme learning machine for subjectivity detection // Journal of the Franklin Institute. 2018. Vol. 355, no. 4. P. 1780–1797. DOI: 10.1016/j.jfranklin.2017.06.007.

20. Ruz G.A., Henríquez P.A., Mascareño A. Sentiment analysis of Twitter data during critical events through Bayesian networks classifiers // Future Generation Computer Systems. 2020. Vol. 106. P. 92–104. DOI: 10.1016/j.future.2020.01.005.

21. Mohammadfam I., Ghasemi F., Kalatpour O., Moghimbeigi A. Constructing a Bayesian network model for improving safety behavior of employees at workplaces // Applied Ergonomics. 2017. Vol. 58. P. 35–47. DOI: 10.1016/j.apergo.2016.05.006.

22. Sierra L.A., Yepes V., García-Segura T., Pellicer E. Bayesian network method for decision-making about the social sustainability of infrastructure projects // Journal of Cleaner Production. 2018. Vol. 176. P. 521–534. DOI: 10.1016/j.jclepro.2017.12.140.

23. McLachlan S., Dube K., Hitman G.A., *et al.* Bayesian networks in healthcare: Distribution by medical condition // Artificial Intelligence in Medicine. 2020. Vol. 107. P. 101912. DOI: 10.1016/j.artmed.2020.101912.

24. Sperotto A., Molina J.-L., Torresan S., *et al.* Reviewing Bayesian Networks potentials for climate change impacts assessment and management: A multi-risk perspective // Journal of Environmental Management. 2017. Vol. 202. P. 320–331. DOI: 10.1016/j.jenvman.2017.07.044.

25. Afenyo M., Khan F., Veitch B., Yang M. Arctic shipping accident scenario analysis using Bayesian Network approach // Ocean Engineering. 2017. Vol. 133. P. 224–230. DOI: 10.1016/j.oceaneng.2017.02.002.

26. Wu J., Zhou R., Xu S., Wu Z. Probabilistic analysis of natural gas pipeline network accident based on Bayesian network // Journal of Loss Prevention in the Process Industries. 2017. Vol. 46. P. 126–136. DOI: 10.1016/j.jlp.2017.01.025.

27. Тулупьев А.Л. Алгебраические байесовские сети: локальный логико-вероятностный вывод. СПб.: Издательство «Анатолия», 2007. 80 с.

28. Тулупьев А.Л. Байесовские сети: логико-вероятностный вывод в циклах. СПб.: Издательство С.-Петербургского университета, 2008. 140 с. Элементы мягких вычислений.

29. Фильченков А.А., Фроленков К.В., Сироткин А.В., Тулупьев А.Л. Система алгоритмов синтеза подмножеств минимальных графов смежности // Информатика и автоматизация. 2013. № 27. С. 200–244. DOI: 10.15622/sp.27.17.

30. Фильченков А.А., Тулупьев А.Л. Алгоритм выявления ацикличности первичной структуры алгебраической байесовской сети по ее четвертичной структуре // Информатика и автоматизация. 2011. № 19. С. 128–145. DOI: 10.15622/sp.19.7.

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

32. Фильченков А.А., Тулупьев А.Л. Структурный анализ систем минимальных графов смежности // Информатика и автоматизация. 2009. № 11. С. 104–129. DOI: 10.15622/sp.11.6.

33. Сироткин А.В., Тулупьев А.Л. Моделирование знаний и рассуждений в условиях неопределенности: матрично-векторная формализация локального синтеза согласованных оценок истинности // Информатика и автоматизация. 2011. № 18. С. 108–135. DOI: 10.15622/sp.18.5.

34. Фильченков А.А., Тулупьев А.Л. Алгоритм выявления ацикличности первичной структуры алгебраической байесовской сети на основе оценки числа ребер в минимальном графе смежности // Информатика и автоматизация. 2012. № 22. С. 205–223. DOI: 10.15622/sp.22.11.

35. Тулупьев А.Л., Сироткин А.В. Локальный апостериорный вывод в алгебраических байесовских сетях как система матрично-векторных операций // Интегрированные модели и мягкие вычисления в искусственном интеллекте. V-я Международная научно-практическая конференция, 9–12 сентября, 2009. Сборник научных трудов. В 2-х т. Т. 1. СПб.: Наука, 2012. С. 425–434.

36. Aho A., Garey M., Ullman J. The Transitive Reduction of a Directed Graph // SIAM Journal on Computing. 1972. Vol. 1, no. 2. P. 131–137. DOI: 10.1137/0201008.

37. Тулупьев А.Л., Сироткин А.В., Николенко С.И. Байесовские сети доверия: логико-вероятностный вывод в ациклических направленных графах. СПб.: Изд-во Санкт-Петербургского ун-та, 2009. 400 с.

38. Веб-приложение по работе с алгебраическими байесовскими сетями. URL: https://abn.dscs.pro/ (дата обращения: 09.03.2023).

39. Автоматизированные алгоритмы АБС, использующие третичную структуру, в частности — глобальный апостериорный вывод. URL: https://abn.dscs.pro/parent_separators_graph (дата обращения: 09.03.2023).

40. Автоматизированные алгоритмы АБС, работающие с первичной структурой, в частности — проверка ацикличности. URL: https://abn.dscs.pro/primary_structure (дата обращения: 09.03.2023).

Вяткин Артём Андреевич, мл. науч. сотр., Санкт-Петербургский федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация)

Абрамов Максим Викторович, к.т.н., руководитель лаборатории теоретических и междисциплинарных проблем информатики, Санкт-Петербургский федеральный исследовательский центр Российской академии наук (Санкт-Петербург, Российская Федерация), старший научный сотрудник

Харитонов Никита Алексеевич, аспирант, кафедра информатики, Санкт-Петербургский государственный университет (Санкт-Петербург, Российская Федерация)

Тулупьев Александр Львович, д.ф.-м.н., проф., профессор, кафедра бизнес-информатики, Северо-Западный институт управления Российской академии народного хозяйства и государственной службы при Президенте Российской Федерации (Санкт-Петербург, Российская Федерация)

DOI: 10.14529/cmse230104

APPLICATION OF TERTIARY STRUCTURE OF ALGEBRAIC BAYESIAN NETWORK IN THE PROBLEM OF A POSTERIORI INFERENCE

© 2023 A.A. Vyatkin[1], M.V. Abramov[1], N.A. Kharitonov[2], A.L. Tulupyev[3]

[1]*Saint Petersburg Federal Research Center of the Russian Academy of Sciences (14th line 39, Vasilievsky Island, St. Petersburg, 199178 Russia),*

[2]*Saint Petersburg State University (Universitetskaya Emb. 7/9, St. Petersburg, 199034 Russia),*

[3]*North-West Institute of Management of the Russian Presidential Academy of National Economy and Public Administration (Sredniy Ave. 57/43, St. Petersburg, 199034 Russia)*

E-mail: aav@dscs.pro, mva@dscs.pro, nak@dscs.pro, alt@dscs.pro

Received: 09.12.2022

In the theory of algebraic Bayesian networks, there are algorithms that allow to conduct a global posterior inference using secondary structures. At the same time, building secondary structures implies the use of tertiary structure. Consequently, the question about the separate application of the tertiary structure in the problem of a posterior inference arises. This issue has been considered earlier, but only a general description of the algorithm has been given, and only models with scalar estimates of the probability of truth have been taken into account. In this paper, we present an algorithm that extends the aforementioned algorithm to the possibility of using it in the case of interval estimates. In addition, an important property of an algebraic Bayesian network is acyclicality, and the correctness of the above-mentioned algorithms is ensured only for acyclic networks. Therefore, it is also necessary to be able to check the acyclicity of an algebraic Bayesian network using a tertiary structure. The description of this algorithm is also presented in this paper, it is based on the previously proved theorem that relates the number of knowledge pattern models in the network to the number of non-empty separators and the number of strong restriction connectivity components in acyclic algebraic Bayesian network, as well as the theorem proved in this paper that two knowledge pattern models belong to the same strong restriction connectivity component. For all the developed algorithms, the correctness of their performance is proved, and their time complexity estimation is calculated.

Keywords: algebraic Bayesian networks, knowledge pattern, logical and probabilistic inference, tertiary structure, probabilistic graphical models, machine learning.

FOR CITATION

Vyatkin A.A., Abramov M.V., Kharitonov N.A., Tulupyev A.L. Application of Tertiary Structure of Algebraic Bayesian Network in the Problem of a Posteriori Inference. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 1. P. 61–88. (in Russian) DOI: 10.14529/cmse230104.

References

1. Larrañaga P., Moral S. Probabilistic graphical models in artificial intelligence. Applied Soft Computing. 2011. Vol. 11, no. 2. P. 1511–1528. DOI: 10.1016/j.asoc.2008.01.003.
2. Yang Y., Xu M., Wu W., *et al.* 3D Multiview Basketball Players Detection and Localization Based on Probabilistic Occupancy. 2018 Digital Image Computing: Techniques and Applications (DICTA). IEEE. 2018. P. 1–8. DOI: 10.1109/DICTA.2018.8615798.

3. Masmoudi K., Abid L., Masmoudi A. Credit risk modeling using Bayesian network with a latent variable. Expert Systems with Applications. 2019. Vol. 127. P. 157–166. DOI: 10.1016/j.eswa.2019.03.014.

4. Qiao W., Liu Y., Ma X., Liu Y. Human Factors Analysis for Maritime Accidents Based on a Dynamic Fuzzy Bayesian Network. Risk analysis. 2020. Vol. 40, no. 5. P. 957–980. DOI: 10.1111/risa.13444.

5. Khlobystova A.O., Abramov M.V., Tulupyev A.L. An Approach to Estimating of Criticality of Social Engineering Attacks Traces. International Conference on Information Technologies, Saratov, February 7–8, 2019. Vol. 199. Springer. 2019. P. 446–456. DOI: 10.1007/978-3-030-12072-6_36.

6. Korepanova A.A., Abramov M.V., Tulupyeva T.V. Identification of User Accounts in the Social Networks "VKontakte" and "Odnoklassniki". Seventeenth Russian Conference on Artificial Intelligence RCAI-2019: collection of scientific papers, Ulyanovsk, October 21–25, 2019. Vol. 2. 2019. P. 153–163. (in Russian).

7. Tulupyev A.L., Nikolenko S.I., Sirotkin A.V. Bayesian Networks: a Logical and Probabilistic Approach. SPb.: Nauka, 2006. 607 p. (in Russian).

8. Tulupyev A.L. Algebraic Bayesian Networks: Global Logical and Probabilistic Inference in Joint Trees. SPb.: Anatolia Publishing House LLC, 2007. 40 p. Elements of Soft Computing. (in Russian).

9. Filchenkov A.A. Minimal join graph set synthesis self-managed possession cliques algorithm. Informatics and Automation. 2010. No. 14. P. 150–169. (in Russian) DOI: 10.15622/sp.14.9.

10. Filchenkov A.A. Minimal join graph set synthesis proprietor possession cliques algorithm. Informatics and Automation. 2010. No. 15. P. 193–212. (in Russian) DOI: 10.15622/sp.15.10.

11. Filchenkov A.A., Tulupyev A.L. The Algebraic Bayesian Network Tertiary Structure. Informatics and Automation. 2011. No. 18. P. 164–187. (in Russian) DOI: 10.15622/sp.18.7.

12. Frolenkov K.V., Filchenkov A.A., Tulupyev A.L. Posteriori inference in tertiary polystructure of an algebraic Bayesian network. Informatics and Automation. 2012. No. 23. P. 343–356. (in Russian) DOI: 10.15622/sp.23.17.

13. Kabir S., Papadopoulos Y. Applications of Bayesian networks and Petri nets in safety, reliability, and risk assessments: A review. Safety Science. 2019. Vol. 115. P. 154–175. DOI: 10.1016/j.ssci.2019.02.009.

14. Amin M.T., Khan F., Ahmed S., Imtiaz S. A data-driven Bayesian network learning method for process fault diagnosis. Process Safety and Environmental Protection. 2021. Vol. 150. P. 110–122. DOI: 10.1016/j.psep.2021.04.004.

15. Baksh A.-A., Abbassi R., Garaniya V., Khan F. Marine transportation risk assessment using Bayesian Network: Application to Arctic waters. Ocean Engineering. 2018. Vol. 159. P. 422–436. DOI: 10.1016/j.oceaneng.2018.04.024.

16. Cai B., Kong X., Liu Y., *et al.* Application of Bayesian Networks in Reliability Evaluation. IEEE Transactions on Industrial Informatics. 2019. Vol. 15, no. 4. P. 2146–2157. DOI: 10.1109/TII.2018.2858281.

17. Wang Z., Chen C. Fuzzy comprehensive Bayesian network-based safety risk assessment for metro construction projects. Tunnelling and Underground Space Technology. 2017. Vol. 70. P. 330–342. DOI: 10.1016/j.tust.2017.09.012.

18. Tavana M., Abtahi A.-R., Caprio D.D., Poortarigh M. An Artificial Neural Network and Bayesian Network model for liquidity risk assessment in banking. Neurocomputing. 2018. Vol. 275. P. 2525–2554. DOI: 10.1016/j.neucom.2017.11.034.

19. Chaturvedi I., Ragusa E., Gastaldo P., *et al.* Bayesian network based extreme learning machine for subjectivity detection. Journal of the Franklin Institute. 2018. Vol. 355, no. 4. P. 1780–1797. DOI: 10.1016/j.jfranklin.2017.06.007.

20. Ruz G.A., Henríquez P.A., Mascareño A. Sentiment analysis of Twitter data during critical events through Bayesian networks classifiers. Future Generation Computer Systems. 2020. Vol. 106. P. 92–104. DOI: 10.1016/j.future.2020.01.005.

21. Mohammadfam I., Ghasemi F., Kalatpour O., Moghimbeigi A. Constructing a Bayesian network model for improving safety behavior of employees at workplaces. Applied Ergonomics. 2017. Vol. 58. P. 35–47. DOI: 10.1016/j.apergo.2016.05.006.

22. Sierra L.A., Yepes V., García-Segura T., Pellicer E. Bayesian network method for decision-making about the social sustainability of infrastructure projects. Journal of Cleaner Production. 2018. Vol. 176. P. 521–534. DOI: 10.1016/j.jclepro.2017.12.140.

23. McLachlan S., Dube K., Hitman G.A., *et al.* Bayesian networks in healthcare: Distribution by medical condition. Artificial Intelligence in Medicine. 2020. Vol. 107. P. 101912. DOI: 10.1016/j.artmed.2020.101912.

24. Sperotto A., Molina J.-L., Torresan S., *et al.* Reviewing Bayesian Networks potentials for climate change impacts assessment and management: A multi-risk perspective. Journal of Environmental Management. 2017. Vol. 202. P. 320–331. DOI: 10.1016/j.jenvman.2017.07.044.

25. Afenyo M., Khan F., Veitch B., Yang M. Arctic shipping accident scenario analysis using Bayesian Network approach. Ocean Engineering. 2017. Vol. 133. P. 224–230. DOI: 10.1016/j.oceaneng.2017.02.002.

26. Wu J., Zhou R., Xu S., Wu Z. Probabilistic analysis of natural gas pipeline network accident based on Bayesian network. Journal of Loss Prevention in the Process Industries. 2017. Vol. 46. P. 126–136. DOI: 10.1016/j.jlp.2017.01.025.

27. Tulupyev A.L. Algebraic Bayesian Networks: Local Logical and Probabilistic Inference. SPb.: Anatolia Publishing House LLC, 2007. 80 p. (in Russian).

28. Tulupyev A.L. Bayesian Networks: Logic-probabilistic Inference in Cycles. SPb.: St. Petersburg University Press, 2008. 140 p. Elements of Soft Computing. (in Russian).

29. Filchenkov A.A., Frolenkov K.V., Sirotkin A.V., Tulupyev A.L. Minimal join graph subsets synthesis system. Informatics and Automation. 2013. No. 27. P. 200–244. (in Russian) DOI: 10.15622/sp.27.17.

30. Filchenkov A.A., Tulupyev A.L. Algorithm for detection of algebraic Bayesian network primary structure acyclicity based on its quaternary structure. Informatics and Automation. 2011. No. 19. P. 128–145. (in Russian) DOI: 10.15622/sp.19.7.

31. Filchenkov A.A., Tulupyev A.L. Connectivity and Acyclicity of the Primary Structure of an Algebraic Bayesian Network. Vestnik of Saint Petersburg University. Mathematics. Mechanics. Astronomy. 2013. No. 1. P. 110–119. (in Russian).

32. Filchenkov A.A., Tulupyev A.L. Minimal joint graph structure synthesis. Informatics and Automation. 2009. No. 11. P. 104–129. (in Russian) DOI: 10.15622/sp.11.6.

33. Sirotkin A.V., Tulupyev A.L. Knowledge and reasoning with uncertainty modeling: matrix-and-vector calculus for local reconciliation of truth estimates. Informatics and Automation. 2011. No. 18. P. 108–135. (in Russian) DOI: 10.15622/sp.18.5.

34. Filchenkov A.A., Tulupyev A.L. Algorithm for Detection Algebraic Bayesian Network Primary Structure Acyclicity Based on Number of Minimal Join Graph Edges Estimating. Informatics and Automation. 2012. No. 22. P. 205–223. (in Russian) DOI: 10.15622/sp.22.11.

35. Tulupyev A.L., Sirotkin A.V. Local Posterior Inference in Algebraic Bayesian Networks as a System of Matrix-and-vector Operations. Integrated Models and Soft Computing in Artificial Intelligence. V-th International Scientific and Practical Conference, September 9–12, 2009. Collection of scientific works. In 2 vols. Vol. 1. SPb.: Nauka, 2012. P. 425–434. (in Russian).

36. Aho A., Garey M., Ullman J. The Transitive Reduction of a Directed Graph. SIAM Journal on Computing. 1972. Vol. 1, no. 2. P. 131–137. DOI: 10.1137/0201008.

37. Tulupyev A.L., Sirotkin A.V., Nikolenko S.I. Bayesian Belief Networks: Logical and Probabilistic Inference in Acyclic Directed Graphs. SPb.: St. Petersburg University Press, 2009. 400 p. (in Russian).

38. Web application for algebraic Bayesian networks. URL: https://abn.dscs.pro/ (accessed: 09.03.2023).

39. Automated ABN algorithms using tertiary structure, in particular – global a posterior inference. URL: https://abn.dscs.pro/parent_separators_graph (accessed: 09.03.2023).

40. Automated ABN algorithms working with the primary structure, in particular – check acyclicity. URL: https://abn.dscs.pro/primary_structure (accessed: 09.03.2023).