Декрементальный синтез третичной структуры алгебраических байесовских сетей[1]

Романов А. В., студент СПбГУ, мл. научный сотрудник СПИИРАН,
exxemka.spb@gmail.com

Аннотация

Рассмотрена проблема изменения третичной структуры при исключении вершины из алгебраической байесовской сети. Третичная структура является одной из глобальных структур АБС и используется как для синтеза, так и для поиска наиболее эффективной вторичной структуры. Предложены декрементальные алгоритмы изменения третичной структуры, вместе с описанием и теоремами о корректности.

Введение

Алгебраическая байесовская сеть представляет собой разновидность графических логико-вероятностных моделей систем знаний с неопределенностью [1]. Исследования на тему АБС направлены на решение двух актуальных проблем: дефицита знаний и дефицита математических моделей для представления таких знаний с неопределенностью [6, 7].

В теории АБС [3, 7] особое внимание уделяют первичной и вторичной структурам. Первичная представляет собой множество фрагментов знаний, характеризующих некую предметную область, а вторичная — определяет взаимосвязи между ними. Вторичную структуру представляют минимальным графом смежности, на котором проводится весь логико-вероятностный вывод [4, 5].

Среди глобальных структур алгебраической байесовской сети также выделяют третичную структуру, которую изначально использовали как вспомогательную для синтеза вторичной [3, 8]. В дальнейших исследованиях область ее применения расширилась до выявления ацикличности [9], построения всего множества минимальных графов смежности [11, 12, 13], а также синтеза случайного минимального графа смежности и поиска наиболее эффективной вторичной структуры [10].

[1]Статья содержит результаты исследований, частично поддержанных грантами РФФИ 15-01-09001-а, 14-01-00580

Проблема синтеза вторичной, а так же третичной структур была рассмотрена в работах [10, 14].

Цель данной работы состоит в том, чтобы предложить декрементальные алгоритмы синтеза третичной структуры, которые изменяют уже существующую структуру при исключении вершины из АБС, а не строят ее заново.

Определения и обозначения

Чтобы сформировать систему обозначений, приведем определения и понятия, уже сложившиеся в [2, 14].

Определение 1. [2] *Графом* $G = (V, E)$ *называется совокупность двух множеств — непустого множества* V *(множества вершин) и множества* E *двухэлементных подмножеств множества* V *(*E *— множество ребер).*

Определение 2. [14] *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*.*

Определение 3. [14] *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, то есть такие, у которых нагрузка любого ребра равна пересечению нагрузок вершин этого ребра.

Определение 4. *Третичная структура — это граф, представленный в виде диаграммы Хассе с порядком следования сверху вниз, построенный на множестве непустых сепараторов.*

Определение 5. *Родителем сепаратора будем называть сепаратор, предшедствующий данному в третичной структуре.*

Определение 6. *Сыном сепаратора будем называть сепаратор, следующий за данным в третичной структуре.*

Декрементальное удаление вершины из третичной структуры

Заданы набор вершин V, множество непустых сепараторов Sep и третичная структура АБС, построенная на этом множестве. Рассмотрим случай удаления вершины из уже построенного графа.

При удалении вершины u есть два варианта.

1. Вершина не содержит в себе уникальных сепараторов,

$$\forall s \in \text{Sep} \;\; W_s \subset W_u \;\; \exists v, k \in V : v \neq u, k \neq u, W_v \cap W_k = W_s.$$

2. Вершина содержит в себе уникальные сепараторы,

$$\exists s \in \text{Sep} \;\; W_s \subset W_u \;\; \nexists v, k \in V : v \neq u, k \neq u, W_v \cap W_k = W_s.$$

В первом случае третичная структура не изменится, так как при удалении вершины множество сепараторов останется неизменным, ведь на каждый сепаратор, входящий в удаляемую вершину, существуют две отличные от удаляемой вершины, в которые он входит. Раз не изменилось множество сепараторов, то не изменится и третичная структура, построенная на этом множестве.

Во втором случае необходимо будет найти сепараторы, которые получены только пересечением нагрузок удаляемой вершины с какой-либо другой, и удалить их из третичной структуры с помощью функции RemoveSepFromThirdStructure, которая будет описана ниже.

Описание алгоритма

На листинге 1 представлен псевдокод изменения третичной структуры при удалении вершины из АБС. На вход подаются третичная структура $G = \langle S, E \rangle$, представленная в виде графа, набор вершин первичной структуры V, вершина u, которая будет удалена, и коллекция сепараторов Seps, представленная в виде словаря, в котором по ключу-сепаратору хранятся значения — сколько раз встречается данный сепаратор при пересечении каждой вершины с каждой.

Для изменения третичной структуры необходимо выделить все непустые сепараторы (строка 5), полученные при пересечении нагрузок данной вершины со всеми нагрузками остальных вершин (строки 3–4) и подсчитать сколько раз они встретились. Для этого будем использовать структуру такого же типа, как и Seps (строки 6–9). Затем

необходимо встречаемость каждого сепаратора sep из множества Seps, который принадлежит также и VertexSeps, уменьшить на соответствующее число из VertexSeps. Если после всех вышеперечисленных операций Seps[sep] будет равняться нулю, что равнозначно тому, что данный сепаратор образуется только пересечением нагрузки удаляемой вершины u с другими вершинами из множества S, то sep больше не будет присутствовать в множестве Seps после удаления вершины, следовательно он не будет присутствовать и в третичной структуре, построенной на множестве непустых сепараторов. Значит, сепаратор sep надлежит удалить из коллекции Seps (строка 14) и третичной структуры (строка 13).

```
   input: G = ⟨S, E⟩, V, u, Seps
   output: G = ⟨S′, E′⟩
 1: function RemoveSeparators
 2:     VertexSeps = ∅
 3:     foreach (v in V)
 4:         sep = W_v ∩ W_u
 5:         if (sep ≠ ∅)
 6:             if (VertexSeps.Contains(sep))
 7:                 VertexSeps[sep] = VertexSeps[sep] + 1
 8:             else
 9:                 VertexSeps.Add(sep, 1)
10:     foreach (sep in VertexSeps)
11:         Seps[sep] = Seps[sep] − sep.Value
12:         if (Seps[sep] == 0)
13:             RemoveSepFromThirdStructure(G, sep)
14:             Seps = Seps.Remove(sep)
15:     return G
```

Алгоритм 1: Декрементальный алгоритм изменения третичной структуры при удалении вершины.

Обоснование корректности алгоритма

Ниже представленно обоснование алгоритма с листинга 1. Описание функции RemoveSepFromThirdStructure и обоснование корректности будет представлено далее.

Теорема 1. *Функция AddSeparators, представленная на листинге 1, при удалении вершины из первичной структуры АБС изменяет третичную структуру так, что полученная структура является третичной для нового набора вершин.*

Доказательство. Рассмотрим подробнее представленный алгоритм. Так как третичная структура построена на непустом множестве сепараторов, а рассматривается именно изменение третичной структуры, то нас интересует только изменение этого множества при удалении вершины. То есть, необходимо найти все сепараторы, которые исключатся из множества непустых сепараторов при удалении вершины, и удалить их из множества сепараторов и третичной структуры. Так как мы перебираем все вершины из множества V'', то в множестве VertexSeps получим все сепараторы, встречаемость которых уменьшится, и число, на которое она будет изменена. Затем в строках 10–11 пересчитаем встречаемость каждого сепаратора из множества сепараторов, и, если встречаемость станет равно 0, то удалим его из множества и из третичной структуры. Таким образом, данный алгоритм корректно изменяет множества сепараторов и третичную структуру при удалении вершины из первичной структуры. Доказательство корректности функции удаления сепаратора RemoveSepFromThirdStructure из третичной структуры предстваленно далее.

Описание алгоритма удаления сепаратора из третичной структуры

При удалении сепаратора sep из третичной структуры, представленной графом $G = (S, E)$ нужно, при необходимости, добавить новые ребра между родителями и сыновьями удаляемой вершины, которые были связаны через нее. Для этого переберем все ребра из множества E (строка 4), и для каждого ребра, которое содержит вершину sep добавим вторую вершину либо в множество сыновей данного сепаратора (строки 5–6), если ребро идет в вершину sep, либо в множество родителей (строки 7–8), если ребро проведено из sep. После этого удалим все ребра, соединяющие родителей с sep (строки 9–10) и sep с сыновьями (строки 11–12). Затем удалим сепаратор sep из множества вершин графа (строка 13) и проверим, необходимо ли соединять родителей с сыновьями ребром, либо между ними уже есть путь в графе через другие вершины (строка 16). Если нет, то соединим их ребром (строка 17).

Обоснование корректности алгоритма

Теорема 2. *Приведенный на листинге 2 алгоритм корректно удаляет из третичной структуры, представленной графом* $G = (S, E)$

```
input: G = ⟨S, E⟩, sep
output: G = ⟨S′, E′⟩
1: function RemoveSepFromThirdStructure
2:     Sons = ∅
3:     Parents = ∅
4:     foreach (edge in E)
5:         if (edge.Source == sep)
6:             Sons = Sons ∪ edge.Target
7:         if (edge.Target == sep)
8:             Parents = Parents ∪ edge.Source
9:     foreach (parent in Parents)
10:        E = E \ {(parent, sep)}
11:    foreach (son in Sons)
12:        E = E \ {(sep, son)}
13:    S = S \ sep
14:    foreach (parent in Parents)
15:        foreach (son in Sons)
16:            if (!PathExists(parent, son))
17:                E = E ∪ {(parent, son)}
18:    return G
```

Алгоритм 2: Алгоритм удаления сепаратора из третичной структуры.

сепаратор sep таким образом, что на выходе получается новый граф, соответствующий третичной структуре, построенной на множестве непустых сепараторов, не содержащего sep.

Доказательство. Чтобы граф $G = (S, E)$, построенный на множестве Sep, являлся третичной структурой, необходимо выполнение следующих условий.

1. Множества Sep и S должны совпадать поэлементно.

2. Ребро между двумя вершинами u и v проведено тогда и только тогда, когда
   $W_u \cap W_v = W_u$ и $\nexists s \in S : \{W_s \cap W_u = W_s$ и $|W_s| > |W_u|\}$.

Так как граф уже построен и является третичной структурой, то первое условие выполненно, следовательно, при удалении сепаратора sep из множеств S и Sep, они не перестанут поэлементно совпадать. Осталось проверить выполнение второго условия после работы алгоритма.

Ребер с одной вершиной не появится, так как в строках 9–12 удаляются все ребра, содержащие вершину sep. Построенные в алгоритме множества Parents и Sons таковы, что

$$\forall p, s \; p \in \text{Parents}, s \in \text{Sons}: \; W_p \cap W_s = W_p.$$

Проверка, что $\nexists k \in V : \{W_k \cap W_s = W_k$ и $|W_p| < |W_k| < |W_s|\}$ проводится в строке 16 (любым алгоритмом поиска пути в графе, например поиском в глубину). Если путь существует, значит такая вершина k существует в графе, значит она уже соединена ребрами с вершинами p и s, следовательно ребро (p, s) добавлять в множество E не нужно. Если же путь не найден, то такой вершины s не существует, а значит вершины p и s должны быть соединены ребром, и это ребро добавляется в строке 17. Таким образом, после работы алгоритма полученный граф является третичной структурой для множества $\text{Sep} \setminus \{\text{sep}\}$.

Заключение

Таким образом, оказалось возможным декрементализировать синтез третичной структуры при исключении вершины из алгебраической байесовской сети. Были разработаны алгоритмы изменения третичной структуры алгебраической байесовской сети при изменении набора ее вершин. Кроме того, для каждого алгоритма был приведен листинг с псевдокодом и теорема о корректности.

Литература

[1] Николенко С.И., Тулупьев А.Л. Вероятностная семантика байесовских сетей в случае линейной цепочки фрагментов знаний // Труды СПИИРАН. — 2005. — № 2. — С. 53–57.

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

[3] Опарин В.В., Тулупьев А.Л. Синтез графа смежности с минимальным числом ребер: формализация алгоритма и анализ его корректности // Труды СПИИРАН. — 2009. — № 11. — С. 142–157.

[4] Тулупьев А.Л. Алгебраические байесовские сети: глобальный логико-вероятностный вывод в деревьях смежности: Учеб. пособие.

СПб.: СПбГУ; ООО Издательство «Анатолия», 2007. 40 с. (Сер. Элементы мягких вычислений).

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

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

[7] Тулупьев А.Л., Сироткин А.В., Николенко С.И. Байесовские сети доверия: логико-вероятностный вывод в ациклических направленных графах. — СПб.: Издательство С.-Петербургского университета, 2009. — С. 400.

[8] Фильченков А.А., Тулупьев А.Л. Структурный анализ систем минимальных графов смежности // Труды СПИИРАН. — 2009. — № 11. — С. 104–127.

[9] Фильченков А.А., Тулупьев А.Л. Анализ циклов в минимальных графах смежности алгебраических байесовских сетей // Труды СПИИРАН. — 2011. — № 17. — С. 151–173.

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

[11] Фильченков А.А. Алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик // Труды СПИИРАН. — 2010. — № 12. — С. 119–133.

[12] Фильченков А.А. Алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик владе ний // Труды СПИИРАН. — 2010. — № 13. — С. 67–86.

[13] Фильченков А.А. Алгоритм построения множества минимальных графов смежности при помощи самоуправляемых клик собственников // Труды СПИИРАН. — 2010. — № 14. — С. 150–169.

[14] Фильченков А.А. Синтез графов смежности в машинном обучении глобальных структур алгебраических байесовских сетей. Дисс. . . к-та физ.-мат. н. Самара, 2013. С. 339.(Самарск. гос. аэрокосм. ун-т им. ак. С. П. Королева (нац. исслед.))