ПРЕДСТАВЛЕНИЕ ЛОКАЛЬНОЙ И ГЛОБАЛЬНОЙ СТРУКТУРЫ АЛГЕБРАИЧЕСКОЙ БАЙЕСОВСКОЙ СЕТИ В JAVA-ПРИЛОЖЕНИЯХ

А. Л. ТУЛУПЬЕВ*, Д. М. СТОЛЯРОВ, М. В. МЕНТЮКОВ

Санкт-Петербургский институт информатики и автоматизации РАН

СПИИРАН, 14-я линия ВО, д. 39, Санкт-Петербург, 199178

<alt@iias.spb.su>

УДК 004.8

*Тулупьев А. Л., Столяров Д. М., Ментюков М. В. Предртя***вление локальной и глобальной структуры алгебраической байесовской сети в Java-приложениях** // Труды СПИИРАН. Вып. 5. — СПб.: Наука, 2007.

Аннотация. *Идеал цепочек конъюнкций с вероятностными оценками истинности рассмотрен как математическая модель фрагмента знаний (ФЗ). Совокупность полученных ФЗ организуется в алгебраическую байесовскую сеть (АБС). Анализируются локальная структура (. .) (. . - ментов знаний и связи между ними) с целью разработки совокупности объектов для представления фрагментов знаний и АБС в программных приложениях, а также для разработки алгоритмов анализа и формирования указанных структур.* — Библ. 16 назв.

UDC 004.8

Tulupyev A. L., Stolyarov D. M., Mentyukov M. V. ***A* Representation for Local and Global Structures of an Algebraical Bayesian Network in Java Applications** // SPIIRAS Proceedings. Issue 5. — SPb.: Nauka, 2007.

Abstract. *A conjunct ideal with probabilistic estimates of its elements is considered as a knowledge pattern (KP) mathematical model. A set of KPs can be assembled in an algebraical Bayesian network (ABN). We analyze local and global structures of ABNs in order to represent KPs and ABNs as object-oriented data structures in software applications. Our primary scope is Java applications.* — Bibl. 16 items.

1. Введение

Целью настоящей работы является описание структур данных, использующихся для представления фрагментов знаний (ФЗ) алгебраических байесовских сетей [3-7] (АБС) в программах на языке Java, а также машин [локального] вывода — объектов, реализующих алгоритмы локального логико-вероятностного вывода (ЛВВ) в ФЗ. Кроме того, помимо вопросов представления первичной структуры АБС (т.е. представления входящих в нее фрагментов знаний без учета связи между ними), предложен подход к построению вторичной структуры АБС, когда связи между ФЗ учитываются и оформляются в виде графа смежности, дерева смежности или линейной цепи фрагментов знаний [5, 7].

2. Математические модели баз фрагментов знаний

Определение 2.01. *Алфавит* — упорядоченное множество атомарных пропозициональных формул (атомов) $A = (x1,\ldots,x_n)$. Алфавит задает лексикографический порядок на множестве атомов.

* Исследования, результаты которых представлены в настоящей работе, были частично под- 2006 2007 .

Предполагается, что предметная область характеризуется высказываниями на языке пропозициональных формул, построенными над атомами и охарактеризованными скалярными (точечными) или интервальными оценками вероятности истинности. Оценка вероятности отражает степень доверия к тому или иному высказыванию.

Определение 2.02. *Аргументное место (литерал)* x, где $x \in A$, может принимать одно из двух значений $\sim x \in \{x, \bar{x}\}$, где $\bar{x}$ — логическое отрицание x.

Определение 2.03. $F = F(A)$ — набор всех возможных пропозициональных формул над A.

Определение 2.04. $F = F(A) = F^{\circ}(A)/=$ — набор классов эквивалентных пропозициональных формул над A.

Определение 2.05. Множество *квантов* $Q = Q(S) = \{ \bigwedge_{x \in S} x \}$, где $S \in A$ и каждое $\sim \in \{x, \bar{x}\}$ (пробегает два своих возможных значения).

Например, $Q = Q(A) = \{x_1 \wedge x_2 \wedge \ldots \wedge \sim_n\}$, где каждое аргументное место «пробегает» два своих возможных значения. Во множестве квантов в рассмотренном случае будет 2^n элементов.

Определение 2.06. *Идеалом цепочек конъюнкций (идеалом конъюнктов)* над заданным множеством $S \in A$ атомарных формул назовем все положительно-означенные цепочки конъюнкций атомарных пропозициональных формул, которые можно образовать над S. Такой идеал обозначим как S°.

Идеал конъюнктов $C(S) = S^{\circ}$ содержит пустой конъюнкт, эквивалентный тождественной истине. Идеал конъюнктов $S^{Л} = (S)$ образуется из S° после исключения пустого конъюнкта.

Отметим, что мы рассматриваем на самом деле *классы* цепочек конъюнкций: мы не различаем цепочки конъюнкций с одинаковым набором аргументных мест и/или атомарных пропозициональных формул.

Над пропозициональными формулами можно определить вероятность истинности; мы рассматриваем подход, предложенный Н. Нильссоном [15, 16], развитый группой Хальперна, Фаджина и Меджидо [13, 14] и адаптированный к теории АБС в [5-11].

Определение 2.07.1. *Математическая модель фрагмента знаний* (knowledge pattern) со *скалярными* (точечными) оценками вероятностей KP — это упорядоченная пара $KP = (S^{\circ}, p)$, где $S \in A$, а p — скалярные (точечные) оценки вероятностей, заданные на элементах идеала p: $S^{\circ} \wedge [0; 1]$.

Определение 2.07.2. *Математическая модель фрагмента знаний* (knowledge pattern) с *интервальными* оценками вероятностей KP — это упорядоченная пара $KP =$, где $S \in A$, а $\mathbf{p}$ — интервальные оценки вероятностей, заданные на элементах идеала $\mathbf{p}$: $S^{\circ} \wedge \{[a;b] | a, b \in [0;1], a < b\}$.

Определение 2.8. Пусть задано k-элементное подмножество атомов из алфавита $S = \{x_{i0}, x^{\wedge}, к, x_{ik\_1}\ x_{i,} \in A,\ 0 < j < k \_ 1,\ i_0 < i_1 < \ldots < i_{k\_1}\}$; сопоставим ему упорядоченную последовательность его элементов S, содержащую эле-

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

менты множества S в обратном лексикографическом порядке $S = (^{x}i_k \quad Xj_1, x_{jo})$. Элементы S индексируются справа налево, счет позиций начинается с нуля; j-й элемент справа записывается как S[j]; в наших обозначениях $S[j] = Xj$.

Заметим, что единица в двоичной записи степени двойки 2^j стоит на той же позиции, что и элемент S[j] в последовательности S. Это несложное наблюдение позволит достаточно просто решить вопрос о построении структур данных для алфавита, его подмножеств, квантов и конъюнктов.

Индексация квантов. Проиндексируем [8, 9, 11] кванты из множества квантов $Q = Q(S)$. Будем считать, что атомы в записи квантов упорядочены в обратном лексикографическом порядке. Сопоставим кванту неотрицательное целое число, в двоичной записи которого на месте отрицательных означиваний атомов стоят нули, а на месте положительных означиваний — единицы. И наоборот, неотрицательному числу в двоичной записи сопоставим квант, у которого на месте нулей стоят отрицательно означенные атомы, а на мести единиц — положительно означенные атомы. Такая индексация задает биекцию между множеством квантов и целыми неотрицательными числами от нуля до $(2^k - 1)$.

Например, для $S = (y_6, У_4, У_2, У_1)$ двоичной записи 1011_2 будет соответствовать квант $y_6 y_4 y_2 y_1$, а 1100_2 — $y_6 y_4 y_2 y$.

Индексация конъюнктов. Проиндексируем [8, 9, 11] конъюнкты из идеала конъюнктов $C = C(S)$. Будем рассматривать конъюнкт как подпоследовательность S . Тогда каждому конъюнкту можно биективно сопоставить двоичную запись числа, содержащую k позиций (на каждой из которых может находиться ноль или единица), как характеристический вектор соответствующей подпоследовательности. Такая индексация задает биекцию между идеалом конъюнктов и целыми неотрицательными числами от нуля до $(2^k - 1$

Например, для $S = (y_6, y_4, y_2, y_1)$ двоичной записи 1011_2 будет соответствовать конъюнкт у6у2yi, а 11002 — у6У4 .

Выбранная индексация задает линейный порядок на квантах и конъюнктах; они упорядочиваются по возрастанию своих индексов.

Векторы вероятностей квантов и конъюнктов. Упорядоченной последовательности квантов сопоставим вектор $\mathbf{Q}^{(k)}$, в котором сверху вниз идут скалярные оценки вероятностей соответствующих квантов. Аналогичным образом упорядоченной последовательности конъюнктов сопоставим вектор $\mathbf{P}^{(k)}$, в котором сверху вниз идут скалярные оценки вероятностей соответствующих конъюнктов [6, 9, 10]. Определим два семейства матриц [6-8, 10] через степень Кронекера [2]:

$$k \quad \begin{matrix} i & -i^{[k]} \\ 0 & i \end{matrix}, \quad \mathbf{J}k \quad \begin{matrix} 1 & 1^{\#]} \\ 0 & 1 \end{matrix} \qquad k^{x J} k = {}^{J}k^{x'} k = {}^{E}k,$$

где $\mathbf{E_k}$ — единичная матрица порядка 2^k

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Оказывается, что справедливы соотношения, играющие решающую роль в определении непротиворечивости оценок во фрагментах знаний и априорном выводе на них: $\mathbf{I_k} \times \mathbf{P}^{(k]} = \mathbf{Q}^{<k}$, $\mathbf{J_k} \times \mathbf{Q}^{(k]} = \mathbf{P}^{<k}$.

Индексация конъюнктов и квантов позволяет хранить сведения об оценках вероятности их истинности в массивах, что существенно облегчает разработку структур данных для представления фрагментов знаний и алгоритмов логико-вероятностного вывода в них.

При обсуждении вопросов структуры для краткости вместо «пересекаются идеалы-носители фрагментов знаний» говорят «пересекаются фрагменты знаний», вместо «идеалы-носители фрагментов знаний имеют общие элементы» — «фрагменты знаний имеют общие элементы» и т.п.; т.е. в указанном контексте фрагменты знаний и их носители отождествляются.

Определение 2.09. *Алгебраической байесовской сетью* (АБС) называется набор фрагментов знаний, в котором ни один фрагмент знаний не входит целиком в другой.

Определение 2.10. Фрагменты знаний, вошедшие в набор, задают *первичную структуру* АБС.

Заметим, что, с точки зрения структуры, фрагмент знаний однозначно задается своим наиболее длинным (главным) конъюнктом. Все остальные конъюнкты в этом ФЗ являются подконъюнктами главного конъюнкта. Таким образом, первичная структура АБС однозначно определяется соответствующим набором главных конъюнктов.

Определение 2.11. Существенную роль в процессе глобального логико-вероятностного вывода в АБС играет структура связей между фрагментами знаний, вошедшими в сеть — *вторичная структура* АБС.

Вторичная структура АБС может быть представлена линейной цепью ФЗ, деревом смежности или графом смежности [6] (их определения будут рассмотрены в следующих разделах).

Алгебраическую байесовскую сеть можно рассматривать как одну из математических моделей баз фрагментов знаний с вероятностной неопределенностью.

3. Представление первичной структуры АБС

Теория АБС — активно развивающая отрасль. Она может быть использована в широком классе задач от биоинформатики до обработки и сжатия изображений. Каждая задача имеет свою специфику и накладывает ограничения на реализацию структуры данных. При построении модели, реализующей АБС, необходимо учитывать множество нюансов. Выбор представления первичной структуры АБС является одной из основных проблем, которые нужно решать еще на этапе проектирования приложения (или комплекса программ).

Первичная структура АБС на локальном уровне может быть реализована множеством способов. Мы остановимся на одном из них и опишем процесс его разработки, а также то, какие трудности при этом встретились и как они были преодолены. Кроме того, мы дадим краткий обзор ядра текущей версии комплекса программ.

Предшествующая реализация первичной структуры АБС [1, 10] показала возможность использования подхода к формированию структур данных для хранения фрагментов знаний, теоретическое описание которого было дано в [5, 6, 9, 11], в программных приложениях. Кроме того, был получен материал для

дальнейших исследований, нацеленных на изучение и совершенствование структур данных и, в целом, архитектуры программного комплекса. Результаты исследований позволили сформировать принципы, которые учитывались при создании следующей версии программного продукта.

В реализации [1, 10] описание фрагментов знаний и свидетельств строилось на интерфейсе Evidence, который являлся корнем иерархии структур хранения и представления данных. Существовали также два вида потомков этого интерфейса: IndeterministicEvidence и DeterministicEvidence для представления недетерминированных и детерминированных свидетельств соответственно. Интерфейс KnowledgePattern наследовал интерфейс IndeterministicEvidence, а в реализующем классе существовал метод априорного вывода.

Для описания функциональности машин апостериорного вывода служил интерфейс Inferrer, а реализующие его классы — LocalScalarInferrer и LocalIntervalInferrer — могли осуществлять апостериорный вывод для ФЗ и свидетельств со скалярными и интервальными оценками соответственно.

При анализе этой модели были обнаружены некоторые недостатки архитектуры.

В частности, классы, реализующие интерфейсы KnowledgePattern и Inferrer, перегружены большим числом методов, что делает код программы малочитабельным, а сопровождение программного продукта — весьма трудоемким. Для решения данной проблемы требуется реализовать некоторые методы в утилитах и отделить логико-вероятностный вывод от классов, представляющих ФЗ.

Такое разделение позволит быстрее передавать объекты, представляющие ФЗ, по сети, что уменьшит трафик. Кроме того, оно также обеспечит хорошую структурированность кода и удобство его чтения для специалистов, занимающихся сопровождением комплекса программ.

Еще одна особенность предыдущей реализации — недостаточно обоснованное разделение в иерархии классов свидетельства и фрагмента знаний. В процессе использования и сопровождения предыдущей версии программного комплекса было замечено, что удобнее было бы представлять свидетельство в виде ФЗ, а не наоборот. В связи с этим появилась необходимость добавить в иерархию интерфейс, описывающий ФЗ с бинарными оценками, а для представления детерминированного свидетельства — использовать реализующий класс этого интерфейса.

Операции априорного вывода интенсивно используются при реализации апостериорного вывода (в частности, это относится к получению оценок вероятности квантов). Следовательно, класс, представляющий машину априорного вывода, должен быть отделен от класса, реализующего машину апостериорного вывода. Более того, первая машина должна использоваться при разработке кода второй.

С помощью объектов этих классов можно распределить задачи между несколькими процессорами или организовать процесс вывода в виде сетевой модели, что обеспечит уменьшение числа операций, производимых каждым отдельным сервером.

Выявленные при анализе особенности классов и их методов позволили сформулировать набор принципов, которые были учтены при реализации новой версии:

- диверсификация классов, реализующих ФЗ, по виду оценок вероятностей (бинарная, скалярная, интервальная);

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

- реализация свидетельства как фрагмента знаний (это становится возможным за счет предыдущей диверсификации классов ФЗ);
- отделение априорного вывода от класса, реализующего ФЗ;
- реализация машин априорного и апостериорного логико-вероятностного вывода как отдельных систем классов и интерфейсов;
- инкапсулирование операций априорного и апостериорного вывода в классы, реализующие машины логико-вероятностного вывода, для обеспечения максимальной независимости от сторонних библиотек[1].

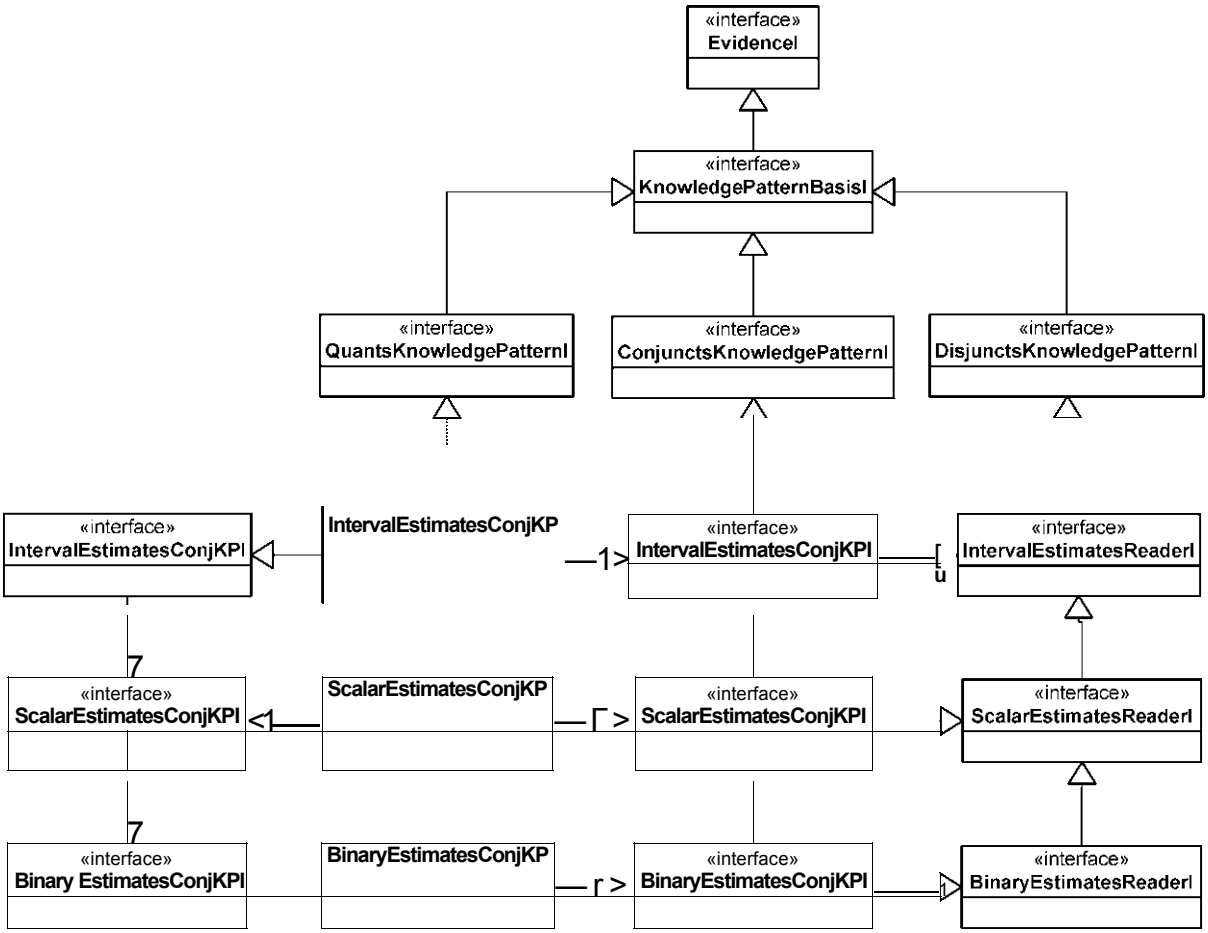


Рис. 2.1. Диаграмма интерфейсов и классов для представления локальных элементов первичной структуры АБС.

Новая реализация модели элементов первичной структуры АБС (рис 2.1) позволяет использовать классы, представляющие фрагменты знаний. Фрагменты знаний отличаются носителем (множество квантов, идеал дизъюнктов, идеал конъюнктов), видом оценок вероятностей, набором операций задания оценок, набором операций обращения к оценкам.

1

работанные платные или бесплатные библиотеки.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Мы рассмотрим ФЗ, носитель которого является идеалом конъюнктов [5, 7]. В зависимости от того, какого вида оценки вероятностей используются, фрагмент знаний может быть представлен одним из трех классов IntervalEstimatesConjKP, ScalarEstimatesConjKP и BinaryEstimatesConjKP. В каждом классе из трех имеются отличающие их друг от друга особенности и методы ввода/вывода.

Хранение свидетельств и фрагментов знаний АБС в одинаковых структурах упрощает операции над этими объектами и делает код более структурированным. Корнем иерархии является интерфейс EvidenceI. Все объекты, которые являются ФЗ, могут использоваться в то же время как свидетельства, поэтому интерфейс KnowledgePatternBasisI — это потомок интерфейса EvidenceI.

Как отмечалось, ФЗ может быть представлен в виде идеала конъюнктов, идеала дизъюнктов, либо набора квантов. Выбор структуры хранения зависит от поставленной задачи и способа ее решения. Предлагаемая модель поддерживает все три вида представления ФЗ (на диаграмме — QuantsKnowledgePatternI, ConjunctsKnowledgePatternI, DisjunctsKnowledgePatternI, которые соответствуют интерфейсам, описывающим представление ФЗ с помощью квантов, идеала конъюнктов и идеала дизъюнктов соответственно), а также утилиты, использующиеся в преобразовании одного представления в другое.

В данной модели добавлены интерфейсы чтения (IntervalEstimatesReaderI, ScalarEsrimatesReaderI, BinaryEstimatesReaderI) и записи (IntervalEstimatesWriterI, ScalarEsrimatesWriterI, BinaryEstimatesWriterI) оценок вероятностей. Использования интерфейсов чтения позволяет не зависеть от конкретного подхода, выбранного для хранения оценок вероятностей элементов фрагмента знаний и доступа к таким оценкам.

Иерархия интерфейсов, описывающих чтение данных, устроена следующим образом (от корня к листьям): IntervalEstimatesReaderI - ScalarEstimatesReaderI - BinaryEstimatesReaderI, поскольку, скажем, бинарные оценки можно прочитать как интервальные, а интервальные как бинарные — нет. Иерархия интерфейсов чтения дает возможность считывания и конвертирования ФЗ с одним типом оценок в ФЗ с другим типом оценок. Или, другими словами, при необходимости ФЗ с бинарными оценками может быть представлен, как ФЗ со скалярными, а ФЗ со скалярными, как ФЗ с интервальными. Следует заметить, что обратное неверно.

Иерархия интерфейсов записи устроена наоборот: BinaryEstimatesWriterI - ScalarEstimatesWriterI - IntervalEsrimatesWriterI. В частности, поясним, что бинарные оценки можно записать во фрагмент знаний, предназначенный для хранения интервальных оценок, а вот интервальные оценки внести в ФЗ, предназначенный для хранения бинарных оценок, нельзя.

Как отмечалось, элементы фрагментов знаний могут иметь бинарные, скалярные или интервальные оценки. Для хранения каждого вида предусмотрен отдельный класс ФЗ. На диаграмме они изображены как BinaryEstimatesConjKP, ScalarEstimatesConjKP, IntervalEstimatesConjKP и являются наследниками интерфейсов чтения и записи соответствующих видов оценок вероятности.

BinaryEstimatesConjKP — класс, реализующий представление фрагмента знаний с бинарными оценками. Данная структура может использоваться для создания тривиального варианта АБС с бинарными оценками, но в основном — для представления и обработки детерминированных свидетельств. Истинностное означивание квантов в таком ФЗ задается выбором одного из квантов над набором указанных атомов. Означивание такого кванта можно хранить как целое число. Истинность конъюнкта определяется значением бита в соответст-

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

вующей позиции двоичного разложения целого числа. Многие операции для ФЗ с бинарными оценками могут быть выполнены с помощью специальных алгоритмов, вычислительная сложность которых меньше, чем у аналогов для ФЗ с другими видами оценок.

ScalarEstimatesConjKP — класс, реализующий представление фрагмента знаний со скалярными оценками. Объекты такой структуры являются компьютерным представлением стохастических свидетельств или фрагментов знаний в АБС со скалярными оценками. Для хранения оценок используется тип double, но значение таких оценок ограничено диапазоном от 0.0 до 1.0.

IntervalEstimatesConjKP — класс, реализующий представление фрагмента знаний с интервальными оценками. С помощью объектов такого типа можно построить любую АБС и представить любое свидетельство, в частности, неопределенное. Операции для объектов этого класса требуют больше ресурсов и имеют большую вычислительную сложность; часть операций использует библиотеки для решения задач линейного программирования. Поэтому такие объекты лучше всего подходят для хранения фрагментов знаний алгебраических байесовских сетей с интервальными оценками или неопределенных свидетельств.

4. Машины локального логико-вероятностного вывода

Общая функциональность машин априорного логико-вероятностного вывода (рис. 4.1) представлена интерфейсом InferrerI. Априорный вывод может выполняться для фрагмента знаний или для всей АБС. Для поддержания глобального вывода в АБС предусмотрен интерфейс GlobalInferrerI (система интерфейсов потомков еще изучается и не будет представлена в данной работе), для поддержания апостериорного вывода в ФЗ — LocalInferrerI. LocalInferrerI и GlobalInferrerI являются потомками InferrerI.

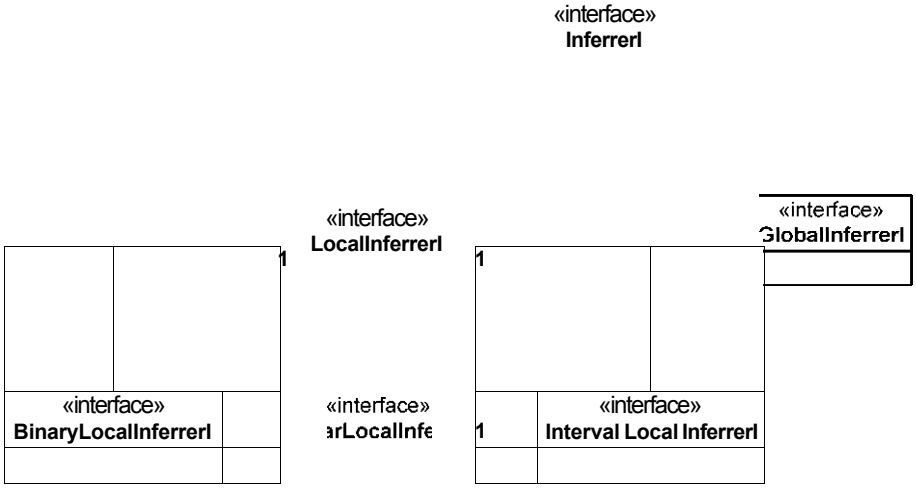


Рис 4.1. Диаграмма интерфейсов, описывающих функциональность машин априорного вывода.

В связи с тем, что поддержание непротиворечивости и априорный вывод зависят от вида оценок, которыми характеризуется вероятность истинности элементов ФЗ, были разработаны отдельные интерфейсы для ФЗ с бинарными, скалярными и интервальными оценками — BinaryLocalInferrerI, ScalarLocalInferrerI, IntervalLocalInferrerI соответственно.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Реализующие классы каждого интерфейса имеют свои особенности.

LocalInferrerI описывает следующие методы:

- boolean processKP() — осуществляет проверку непротиворечивости;
- boolean isConsistent() — возвращает сведения о совместимости оценок;
- boolean isNarrowed() — возвращает сведения о том, что оценки в результате вывода были уточнены;

BinaryLocalInferrerI описывает машину априорного вывода в ФЗ с бинарными оценками и определяет следующие методы:

- QuantsKnowledgePatternI getQuant();
- QuantsKnowledgePatternI getQuant(int substitution);

ScalarLocalInferrerI описывает машину априорного вывода в ФЗ со скалярными оценками и содержит следующие методы вывода результатов:

- ScalarEstimatesConjKPI getSCKP() — возвращает ФЗ с уточненными оценками;
- ScalarEstimatesQuantsKPI getSQKP(int substitution) — возвращает вероятности квантов в виде ФЗ, построенного над квантами;
- ScalarEstimatesConjKPI getSignSubstitutionSCKP(int substitution) — возвращает ФЗ с переозначенными конъюнктами и пересчитанными вероятностями;

IntervalLocalInferrerI описывает машину априорного вывода в ФЗ с интервальными оценками и содержит следующие методы:

- IntervalEstimatesConjKP getRoconciledICKPO — возвращает ФЗ с уточненными оценками;
- IntervalEstimatesQuantsKP getReconciledIQKP() — возвращает ФЗ с уточненными оценками вероятностей квантов;
- IntervalEstimatesConjKP getSignSubstitutionICKP(int substitution) — возвращает ФЗ с переозначенными конъюнктами и пересчитанными оценками вероятностей.

Заметим, что апостериорный вывод в реализующем классе соответствующей машины вывода многократно обращается, в частности, к результатам априорного вывода для ФЗ со скалярными оценками, поэтому следует подчеркнуть важность разработки эффективного алгоритма априорного вывода для ФЗ со скалярными оценками.

В корне иерархии интерфейсов (рис. 4.2), описывающих апостериорный вывод, находится PropagatorI. Апостериорный вывод, также как и априорный, может быть локальный и глобальный: GlobalPropagatorI и LocalPropagatorI — интерфейсы, представляющие функциональность машин глобального и локального апостериорного вывода.

BinaryLocalPropagatorI, ScalarLocalPropagatorI, IntervalLocalPropagatorI являются потомками интерфейса LocalPropagatorI и разработаны для апостериорного вывода в ФЗ с бинарными, скалярными и интервальными оценками соответственно.

На выбор машины локального апостериорного вывода влияет вид поступающего свидетельства — является ли оно детерминированным, стохастическим, неопределенным.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

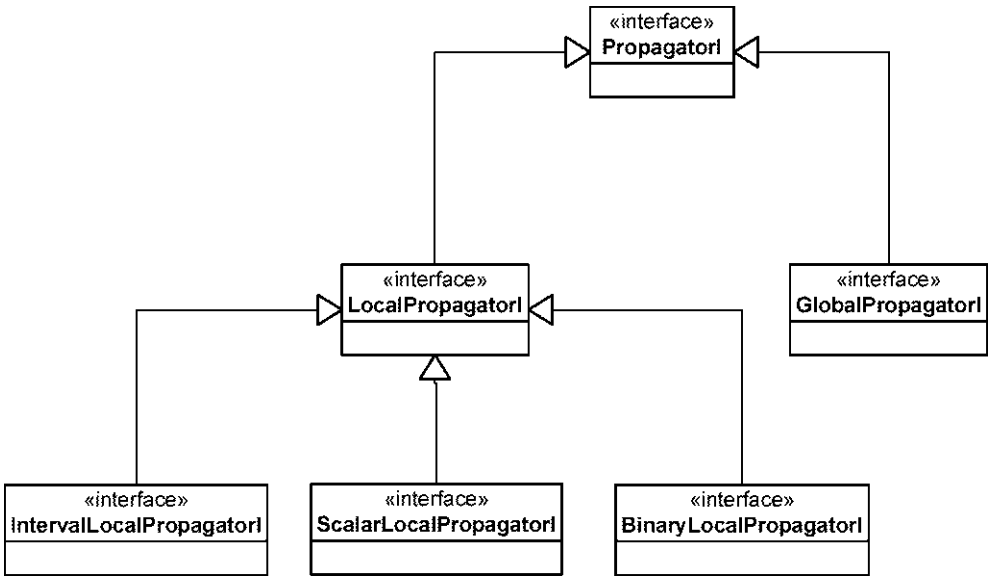


Рис 4.2. Диаграмма интерфейсов, описывающих функциональность машин

.

BinaryLocalPropagatorI

- boolean setKnowledgePattern(BinaryEstimatesConjKPI knowledgePattern) — загружает фрагмент знаний с бинарными оценками;
- boolean setEvidence(BinaryEstimatesConjKPI evidence) — загружает свидетельство, представленное в виде ФЗ с бинарными оценками;
- int getEvidenceProbability() — возвращает вероятность свидетельства над ФЗ;
- BinaryEstimatesConjKPI getBinaryEstimatesConjKPAfterPropagation() — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ с бинарными оценками;
- BinaryEstimatesConjKPI getBinaryEstimatesConjSubKPAfterPropagationO — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ с бинарными оценками и не имеющий общих элементов со свидетельством;
- IntervalEstimatesConjKPI getIntervalEstimatesConjKPAfterPropagation() — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ с интервальными оценками;
- IntervalEstimatesConjKPI getIntervalEstimatesConjSubKPAfterPropagation() — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ с интервальными оценками и не имеющий общих элементов со свидетельством.

ScalarLocalPropagatorI

- double getEvidenceProbability() — возвращает вероятность свидетельства над ФЗ;
- boolean setKnowledgePattern(ScalarEstimatesConjKPI knowledgePattern) — загружает фрагмент знаний с интервальными оценками;
- boolean setEvidence(BinaryEstimatesConjKPI evidence) — загружает свидетельство, представленное в виде ФЗ с бинарными оценками;
- boolean setEvidence(ScalarEstimatesConjKPI evidence) — загружает свидетельство, представленное в виде ФЗ со скалярными оценками;

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

- ScalarEstimatesConjKPI getScalarEstimatesConjKPAfterPropagationO — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ со скалярными оценками;
- ScalarEstimatesConjKPI getScalarEstimatesConjSubKPAfterPropagation() — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ со скалярными оценками и не имеющий общих элементов со свидетельством;
- IntervalEstimatesConjKPI getIntervalEstimatesConjKPAfterPropagation() — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ с интервальными оценками;
- IntervalEstimatesConjKPI getIntervalEstimatesConjSubKPAfterPropagation() — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ с интервальными оценками и не имеющий общих элементов со свидетельством.

IntervalLocalPropagatorI

- double getEvidenceProbabilityLB() — возвращает нижнюю оценку вероятности свидетельства над ФЗ;
- double getEvidenceProbabilityUBO — возвращает верхнюю оценку вероятности свидетельства над ФЗ;
- boolean setKnowledgePattern(IntervalEstimatesConjKPI knowledgePattern) — загружает ФЗ с интервальными оценками;
- boolean setEvidence(BinaryEstimatesConjKPI evidence) — загружает свидетельство, представленное в виде ФЗ с бинарными оценками;
- boolean setEvidence(ScalarEstimatesConjKPI evidence) — загружает свидетельство, представленное в виде ФЗ со скалярными оценками;
- boolean setEvidence(IntervalEstimatesConjKPI evidence) — загружает свидетельство, представленное в виде ФЗ с интервальными оценками;
- IntervalEstimatesConjKPI getIntervalEstimatesConjKPAfterPropagation() — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ с интервальными оценками;
- IntervalEstimatesConjKPI getIntervalEstimatesConjSubKPAfterPropagation() — возвращает набор полученных апостериорных оценок, представленный в виде ФЗ с интервальными оценками и не имеющий общих элементов со свидетельством.

5. Построение вторичной структуры АБС

Первичная структура алгебраической байесовской сети — это та совокупность всех максимальных фрагментов знаний, которые в указанную сеть входят. Заметим, что по определению первичной структуры, ни один фрагмент знаний не может полностью входить в любой другой фрагмент знаний; однако некоторые (или даже все) фрагменты знаний могут иметь общие элементы. Обратим также внимание на то, что непустое пересечение идеалов-носителей ФЗ является идеалом; таким образом, в определенном смысле непустое пересечение фрагментов знаний является в свою очередь фрагментом знаний.

То, что фрагменты знаний могут пересекаться, т.е. их идеалы-носители могут иметь общие элементы, существенно влияет на все процессы логико-

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

вероятностного вывода в алгебраической байесовской сети. Структуру связей между фрагментами знаний, которая выявляется при рассмотрении их общих элементов, необходимо изучить. Такой структуре требуется, по возможности, дать строгое описание, еще лучше — представление в виде математического объекта, свойства которого можно затем изучить математическими методами и представление которого в виде структур данных в компьютерной программе можно разработать.

Структура связей между фрагментами знаний, возникающая за счет их общих элементов, называется *вторичной структурой* алгебраической байесовской сети. Целью раздела является построение вторичной структуры АБС в виде графа смежности (а, в частных случаях, дерева смежности или линейной цепи) фрагментов знаний; причем при таком построении желательным является минимальность графа смежности по числу ребер, в него входящему, а также отсутствие в получаемом графе циклов (т.е. при формировании вторичной структуры АБС более желательно получить не просто граф, а дерево смежности). Ациклические АБС (т.е. АБС со структурой дерева смежности) обладают рядом особых свойств, позволяющих использовать более простые алгоритмы логико-вероятностного вывода [6-8].

Структура каждого отдельного фрагмента знаний однозначно определяется содержащимся в нем конъюнктом наибольшей длины (главный конъюнкт). Все остальные элементы ФЗ — подконъюнкты такого конъюнкта, включая пустой, эквивалентный тождественной истине. Наличие непустых общих элементов между двумя фрагментами знаний взаимооднозначно определяется наличием общих атомов в главных конъюнктах этих ФЗ. Таким образом, вторичная структура АБС зависит лишь от набора множеств атомов, формирующих главные конъюнкты фрагментов знаний, содержащихся в алгебраической байесовской сети. Поэтому в данном разделе представляется целесообразным использовать язык множеств для изложения полученных результатов и алгоритмов построения (точнее, *восстановления)* вторичной структуры АБС по ее первичной структуре.

Для удобства введем ряд *адаптированных* соглашений и обозначений.

Определение 5.01. *Алфавитом* называется множество атомарных пропозициональных формул $A = \{,\ldots, x_n\}$.

Структура и состав идеала-носителя фрагмента знаний однозначно определяется тем подмножеством атомов V е A, из которого сформирован главный конъюнкт самого идеала. Следовательно, для задания первичной структуры АБС достаточно перечислить все такие подмножества алфавита A, из которых сформированы главные конъюнкты фрагментов знаний, вошедших в АБС.

Определение 5.02. Множество подмножеств алфавита **V** = {V е A}-=m перечисляет наборы атомов, вошедших в главные конъюнкты максимальных ФЗ заданной алгебраической байесовской сети.

Далее мы исключаем из рассмотрения несобственные подмножества алфавита $V = A$ и $V = 0$. В первом случае окажется, что все остальные ФЗ будут входить в ФЗ, образованный над идеалом A; во втором случае получится ФЗ над 0°, содержащий лишь пустую конъюнкцию, что неинформативно.

Кроме того, для любых i Ф j справедливо, что V £ Vj и Vj £ V, т.е. ни одни элемент **V** не входит в другой. Иначе у нас в АБС были бы фрагменты знаний, которые бы полностью входили бы в другие фрагменты знаний.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Нашей целью является построение вторичной структуры АБС, заданной над **V** , для этого введем определение графа и дерева смежности.

Определение 5.03. *Множеством вершин* графа смежности является ранее определенное множество **V**. Обозначения V при $1 < i < m$ рассматриваются как метки вершин графа смежности.

Определение 5.04. *Весом* $W(V)$ вершины V графа смежности является множество атомов из алфавита, вошедших в V .

Фактически, $W(V) = V$, но мы сохраним различие в обозначениях для большей выразительности, хотя достаточно понимать, что метками и весами вершин графа смежности будут одни и те же множества из **V** .

Определение 5.05. *Граф смежности G* — это *ненаправленный* граф, заданный парой множеств (**V,E),** где **V —** *множество вершин* (Vertex), определенное ранее, а **E —** *множество ребер* (Edge). При этом должны выполняться дополнительные условия:

1) между каждой парой несовпадающих узлов V и V_j, веса которых содержат общие элементы $W(V) \cap W(V_j) \neq 0$, существует путь [2];
2) в веса каждого из узлов пути, указанного в предыдущем пункте, входят все элементы, общие для весов начального и конечного узлов;
3) вес одного узла графа не входит полностью в вес никакого другого узла графа.

Определение 5.06. *Вес ребра* $W(\{V, V_j\})$, $\{V_i, V_j\} \in$ **E**, графа смежности *G* определяется как пересечение весов тех вершин, которые соединены этим ребром $W(\{V, V_j\}) = W(V) \cap W(V_j)$.

Примем соглашение о том, что ребра с пустым весом в граф смежности не входят.

Определение 5.07.1. *Дерево смежности T* — это граф смежности без циклов (ациклический граф смежности).

Ввиду важности понятия дерева смежности дадим его развернутое определение.

Определение 5.07.2. *Дерево смежности T* — это ненаправленный граф, заданный парой множеств (**V,E),** где **V —** *множество вершин* (Vertex), определенное ранее, а **E** — *множество ребер* (Edge). При этом должны выполняться дополнительные условия:

1) между каждой парой несовпадающих узлов V_i и V_j , веса которых содержат общие элементы $W(V_i) \cap W(V_j) \neq 0$, существует путь;
2) веса узлов любого пути, начинающегося в V_b и заканчивающегося в V_e , содержат пересечение весов этих узлов $W V) \cap W(V_e)$;

[2] Заметим, что этот путь может содержать одно ребро или несколько ребер, а узлы с весами, содержащие общие элементы, не обязательно соединены ребром напрямую — достаточно, если они соединены путем. Подчеркнем также, что рассматриваются пути без самопересечений.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

3) вес одного узла дерева не входит полностью в вес никакого другого узла дерева;
4) любые два узла соединены не более чем одним путем[3].

Определение 5.08. *Линейная цепь* (или просто *цепь)* фрагментов знаний — это дерево смежности, в котором можно выделить два узла V_b и V_e, между которыми существует путь (без самопересечений), причем этот путь содержит все оставшиеся узлы дерева смежности.

Следует обратить внимание на то, что узлы V_b и V_e могут и не содержать общих элементов.

Далее, как правило, будем все графы смежности, заданные над одним и тем же множеством вершин **V**, считать частично упорядоченными по отношению вхождения множеств их ребер **E**: G G", если соответствующие множества ребер **E'** с **E''**. Поскольку множество вершин для всех графов смежности совпадает по нашему соглашению, графы смежности на заданном множестве вершин равны тогда и только тогда, когда совпадают множества их ребер.

Наша ближайшая цель — описать алгоритм для построения хоть какого-то графа смежности на заданном множестве вершин **V**.

На самом деле, наиболее просто строится G_{max} — наибольший (напомним — наибольший по числу ребер) граф смежности на заданном множестве вершин.

G_{max} интересен тем, что содержит все остальные графы смежности; более точно — множество ребер любого графа смежности (в том числе и минимального) будет подмножеством множества ребер графа G_{max}. Следовательно, минимальные графы смежности можно, в частности, получить из G_{max} исключением части ребер. Кроме того, для дальнейших рассуждений и построений нам потребуются клики (удовлетворяющие определенным требованиям, а не все) наибольшего графа смежности G_{max}.

Принципы построения G_{max}. Изначально задано множество вершин **V** графа G_{max}. Множество его ребер пусто. Проведем ребро (т.е. добавим его во множество ребер графа) внутри каждой пары вершин, пересечение весов которых не пусто. Граф построен, причем он удовлетворяет определению графа смежности — каждая пара вершин, веса которых имеют непустое множество общих элементов (т.е. пересекаются), соединена ребром, а значит, и путем.

Далее мы будем искать признаки (или необходимые и достаточные условия) того, что граф смежности минимален. Для этого нам потребуется дополнительный понятийный аппарат, который поможет «систематизировать» веса вершин и ребер, точнее, рассматривать подграфы с менее богатым набором весов, чем заданный изначально. Для этого будем использовать операцию сужения графа.

Определение 5.09. Сужение $G|U$ ненаправленного графа G на подмножество U алфавита A — это ненаправленный граф, в который входят только те вершины и ребра исходного графа G, веса которых содержат U.

[3] Можно воспользоваться альтернативной формулировкой этого условия: любые два узла соединены одним и только одним путем. А те деревья, которые изначально распадаются на отдельные компоненты связности, рассматривать покомпонентно.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

В первую очередь исследуем сужение $G_{max}\,IU$ максимального графа смежности G_{max} на произвольное непустое подмножество U алфавита A.

Заметим, что если $G_{max}\,IU$ не оказался пустым, то он является кликой (т.е. полным графом — в нем любая вершина соединена со всеми оставшимися). В этом случае все вершины, попавшие в сужение $G_{max}\,IU$, имеют в весе общие символы (все символы из того самого множества U), а по построению максимального графа смежности G_{max} любая пара вершин, имеющая непустое пересечение весов, должна быть соединена ребром. Таким образом, действительно получается клика.

Сужение $G_{max}\,IU$ может оказаться пустым, если не найдется вершин из **V**, веса которых включают в себя U, или одноэлементным, если не найдется ребер из **E**, веса которых включают в себя U.

Определение 5.10. *Весом* W(Clique^) *клики* Clique^ графа смежности G является пересечение всех весов вершин, попавших в эту клику. Иначе говоря, *вес клики* — это наибольшее по включению множество атомов из алфавита, которое входит в веса всех вершин клики.

Клика P содержит клику Q тогда и только тогда, когда вес клики P входит в вес клики Q ($W(P)$ с W(Q)).

Далее для краткости будем иногда говорить «*сужение на (под)клику P*», подразумевая «*сужение на вес W (P) (под)клики P*».

К сужению $G_{max}\,IU$ можно снова применить операцию сужения, полученный результат — снова сузить, и т.д. Всякий раз, когда в результате очередного сужения будет получаться непустой и неодноэлементный граф, он будет подкликой клики, образовавшейся на предыдущем шаге, и эта подклика будет также являться кликой в исходном графе G_{max}. Зададимся задачей в явном виде представить структуру отношений между получающимися в результате последовательности сужений кликами и подкликами — построим *направленный* граф клик.

Подчеркнем, что нас интересуют множество всех подклик (включая максимальные клики) **Clique** графа G_{max}, которое

- не содержит пустых подклик,
- не содержит одноэлементных подклик,
- не содержит подклик с одинаковым весом,
- не содержит подклик с пустым весом.

Иначе говоря, множество **Clique** состоит из непустых неодноэлементных подклик с непустыми различающимися весами.

Определение 5.11. Граф клик C — это направленный граф, вершинами которого являются клики из множества **Clique**, содержащиеся в графе G_{max}. Ребра в графе клик C проведены из вершины P в вершину Q, когда клика из вершины P содержит клику из вершины Q. Весом вершины в графе клик является вес соответствующей ей (под)клики.

Направленных циклов в графе клик C нет, поскольку в любом направленном пути в графе клик C уменьшается число вершин графа смежности в вершине-(под)клике.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Заметим, что максимальные клики в ненаправленном графе-носителе графа клик C не связаны друг с другом непосредственно; они могут быть связаны через своих потомков ненаправленным путем. Заметим, что граф клик может быть несвязным.

Теперь перейдем к исследованию особенностей минимальных графов смежности (напомним — минимальность по отношению вхождения множеств ребер) с целью установить общие свойства их структуры. Эти свойства позволят нам в свою очередь описать работу алгоритма построения всех минимальных графов смежности.

Пусть M — произвольный минимальный граф смежности, построенный над заданным множеством вершин $\mathbf{V}$; таким образом, он является подграфом максимального графа смежности G_{max}.

Чтобы изучить особенности структуры выбранного минимального графа смежности M, рассмотрим его сужение MIU на произвольное непустое множество элементов алфавита U. Если MIU непусто и неодноэлементно, то оно будет одним из минимальных графов смежности для вершин из клики-сужения $G_{max}\ IU$. Минимальность $M\ IU$ основана на справедливости следующего наблюдения. Если бы из MIU можно было исключить ребро, не нарушая при этом условий из определения графа смежности, то и из исходного M можно было бы удалить ребро; но тогда исходный граф смежности M не был бы минимальным, а его минимальность предполагалась изначально.

Заметим, что получившийся в результате сужения граф смежности MIU связен. Все вершины этого графа смежности имеют общие элементы, поскольку MIU является сужением графа смежности M на непустое множество элементов алфавита U. А поскольку MIU — граф смежности, это гарантирует, что между любыми его двумя вершинами, вес которых содержит общие элементы, существует путь, что и обеспечивает связность MIU.

Определение 5.12. Граф $M \wedge U$, где $U \subset A$, образуется из графа MIU исключением из него ребер, вес которых *совпадает* с U, и называется *строгим сужением* M на U.

Заметим, строгое сужение оказывается, как правило, сформированным из тех вершин и ребер графа смежности, вес которых содержит U как *собственное* подмножество. Исключением является случай, когда U совпадает с весом какой-нибудь вершины графа смежности. Тогда строгое сужение состоит только из этой вершины. Если бы нашлось хоть одно ребро с весом, содержащим U, то тогда вес указанной вершины полностью входил бы в вес другой вершины, что запрещено по определению графа смежности. Другим исключением является пустое строгое сужение.

Если же U в строгом сужении $M \wedge U$ совпадает с весом ребра графа смежности, то тогда вершины, соединенные ребром, в строгое сужение попадают, а само ребро — нет.

Заметим, что любое ребро из строгого сужения лежит в некоторой собственной (несовпадающей со всем $G_{max}\ IU$ и непустой) подклике $G_{max}\ IU$, хотя бы в подклике с весом этого ребра. Так же заметим, что пересечение MIU с любой такой подкликой лежит в $M \wedge U$. Объясним, почему эти два замечания справедливы. Любое ребро из строгого сужения имеет вес U', который является собственным надмножеством U, т.е. $U' = U \cup B$, где $B \subset A$, $B \cap U = 0$ и

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$B \wedge 0$. Это ребро соединяет две вершины, веса которых в свою очередь являются собственными надмножествами U' (если бы вес вершины совпал бы с весом ребра, то вес такой бы вершины целиком бы вошел в вес другой, а такое не допускается по требованиям к исходным данным). Веса всех вершин и ребер подклики $G_{max}\ IU'$ содержат U, но не совпадают с ним. Соответственно, пересечение MIU и $G_{max}\ IU'$, содержит только те вершины и ребра из MIU, веса которых в свою очередь содержат U, но не совпадают с ним; а значит, это пересечение лежит в $M \wedge U$ по определению строгого сужения.

Отсюда следует, что граф $M \wedge U$ можно представить как объединение[4] графов-сужений $(M\ I\ U)\ W(Q)$, где Q пробегает все множество собственных неединичных подклик (с непустым весом) клики $G_{max}\ IU$; но из этого множества исключаются клики, полностью покрывающие MIU. Иначе говоря, Q пробегает все множество подклик-сыновей клики $G_{max}\ IU$ в графе клик C. Еще один способ описания — Q пробегает все множество подклик, к которым в графе клик C идет направленное ребро из клики $G_{max}\ IU$.

Заметим, что любое такое сужение $(M\ IU)\ W(Q)$ связно: так как любые две вершины из него имеют общий подвес в исходном графе смежности M, между ними есть путь (из определения графа смежности), а этот путь целиком лежит в сужении $(M\ IU)\ W(Q)$.

Определение 5.13. $K(S)$ — множество компонент связности графа S.

Исследуем структуру множества $K(M \cup U)$ — фактически, мы будем изучать структуру графа-сужения MIU, у которого исключены ребра с весом, равным U.

Оказывается, множество $K(M \cup U)$ компонент связности не зависит от выбора изначального минимального графа смежности M над заданным множеством вершин **V**.

Идея доказательства состоит в том, что надо посмотреть на клики графа G_{max}, исследовать компоненты связности его сужения $G_{max}\ IU$ на подклику с весом U при исключении ребер с весом U и подметить, что множества $K(M \cup U)$ и $K(G_{max} \wedge U)$ совпадают ввиду того, что любое сужение на вес подклики связно, как было сказано раньше. Теперь приведем более подробные рассуждения, т.е. докажем, что наборы компонент связности $K(M \cup U)$ и $K(G_{max} \wedge U)$ совпадают. Тогда и для всех минимальных графов смежности M наборы компонент связности $K(M \cup U)$ совпадают[5].

4

5 , .

набор компонент связности графа. Каждой компоненте связности сопоставим множество вершин, в ней содержащееся. Получим набор множеств вершин. Будем говорить, что наборы компонент связности двух или более графов совпадают тогда и только тогда, когда соответствующие наборы множеств вершин совпадают. Дело в том, что взятые из двух разных графов компоненты связности с одинаковым набором множеств вершин могут не совпадать как графы — у них может быть разный набор ребер. Но нас интересует именно множество вершин, содержащееся в компоненте связности.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Для доказательства этого факта достаточно принять во внимание, что для любой вершины V' компонента связности LL, ее содержащая (для каждой вершины такая компонента связности всегда ровно одна), в $M \wedge U$ совпадает как множество вершин с компонентой связности L, содержащей V', из $G_{max} \vee U$. А для этого достаточно проверить, что L и L содержатся друг в друге. Будем проверять каждое включение путем применения определения компоненты связности — множества вершин, до которых можно проложить путь из V' в соответствующем графе.

Проверим, что L содержит L (необязательно собственным образом). Это очевидно: если в некоторая вершина V'' лежит в L, то в $M \wedge U$ есть путь h между V'' и V'. А так как граф $G_{max} \vee U$ содержит все ребра $M \wedge U$, то в $G_{max} \vee U$ вершины V'' и V' тоже соединены тем же путем h. Значит, L содержит L^{lo}.

В обратную сторону включение проверяется не столь просто. Для этого нам потребуется воспользоваться тем, что сужение любого графа смежности на любую подклику связно. Опять же, пусть вершина V'' соединена с вершиной V' в $G_{max} \vee U$ посредством пути h. Ввиду того, что $G_{max} \vee U$ есть объединение клик (потомков в графе клик клики, отвечающей весу U) h можно представить как объединение k подпутей hj, $1 < j < k$, таких, что, во-первых, начало подпути hi совпадает с началом всего пути h, во-вторых, конец последнего подпути hk совпадает с концом всего пути h, в-третьих, для всех j, которые $1 < j < (k-1)$, конец hj и начало $hj+1$ совпадают, в четвертых, каждый отдельный hj полностью помещается в одной подклике, в-пятых, для всех j, которые $1 < j < (k-1)$, hj и $hj+i$ лежат в разных подкликах. Это утверждение легко доказать по индукции по длине пути h. Обозначим эту длину для ясности той же буквой j.

База индукции ($j = 1$). Очевидно, при $j = 1$ справедливо $h1 = h$.

Индукционный переход (от $j-1$ к j). Пусть Y — предпоследняя вершина пути. Тогда путь h', который получается из h исключением последней вершины (т.е. вершины V''), имеет длину $(j-1)$, следовательно, к нему применимо предположение индукции. Пусть для него есть набор из k подпутей h'. Тогда рассмотрим два случая. Первый: hk и V'' лежат в одной компоненте связности. Тогда для h сохраняется тот же набор подпутей h', лишь последний подпуть становится длиннее на одно ребро и одну вершину — к hk в конец присоединяется вершина V''. Второй случай: hk и V'' лежат в разных компонентах связности. Тогда k увеличивается на 1, а в набор подпутей h' добавляется новый $(k+1)$-й подпуть $hk+1$ — путь, состоящий из ровно одного ребра между вершиной Y (концом hk) и вершиной V''.

Индукционный переход осуществлен, утверждение доказано.

Теперь вспомним, что мы хотели построить такой путь hM в $M \cup U$, что его начало и конец — вершины V' и V'' соответственно. Будем строить его опять

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

же при помощи метода математической индукции по k — числу подпутей из построенного разбиения.

База индукции (k = 0): очевидна.

Индукционный переход от k к (k +1).

Действительно, пусть для пути hM в разбиении (k +1) подпуть. Тогда обозначим конец подпути $hM\,k$ как Y. Заметим, что для пути hM = (м \ hk+i)u|Y} (то есть для пути из V' в Y, идущего по ребрам hM) верно предположение индукции. Значит, в M UU существует hM с началом в V' и концом в Y. Заметим, что по построению подпути hk+i вершины Y и V'' лежат в G_{max} Ш U в одной подклике Q. Тут мы вспоминаем, что сужение M U U на любую подклику связно. Значит, в сужении M U U на подклику Q есть путь hY из Y в V''. Тогда путь $(hM$ и $hY)$ соединяет в M^U вершины V' и V''. Индукционный переход доказан.

Значит, вершина V'' лежит в компоненте связности L . Тогда L содержит L. Мы доказали, что $L = L$.

Между любыми двумя элементами $K(M$ U $U)$ (т.е. между любыми двумя несовпадающими компонентами связности из графа M ^ U) в графе смежности MIU есть ровно один путь, который формируется из части или из всех исключенных ребер, вес которых совпадает с U . (Так как MIU связен, компоненты связности, получающиеся после исключения из него ребер с весом U, соединены в MIU ребрами веса U .)

Никакие две такие компоненты не соединены двумя ранее исключенными ребрами, вес которых совпадает с U . Это справедливо в силу минимальности графа смежности MIU, так как если есть два ребра g и h, соединяющие компоненты связности A и B из $K(M$ U $U)$, то после исключения g из M, любой путь из определения графа смежности, проходящий через g, можно заменить путем, проходящим через A - h - B, поскольку A и B связны и все их ребра содержат вес g (который по определению M ^ U, равен U), а также вес h равен весу g (тоже по определению M ^ U).

Следовательно, G_{max} IU — «дерево» на компонентах связности $K(M$ U $U)$, в этом «дереве» все ребра имеют вес, равный U .

Выскажемся более строго. Пусть Y — граф, вершины которого соответствуют элементам $K(M$ U $U)$, а ребро между двумя таким элементами проводится, если они в MIU соединены ребром (не более чем единственным по предыдущему пункту). Тогда такой граф Y — дерево, так как он связный, и если есть цикл, то надо рассмотреть ребро, соответствующее одному из его ребер в G_{max} ^ U (тут опять же важно, что компоненты соединены в G_{max} IU не более, чем 1 ребром), и исключить его, тогда условие графа смежности не нарушится, что противоречит минимальности M.

Мы установили структуру минимального графа смежности M: сужение M на любое подмножество атомов U е A есть дерево на компонентах связности объединения сужений на все такие надмножества множества U, что являются весами клик, в которые в графе клик C идет ребро из клики с весом U. Также

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

можно заметить, что если мы знаем, как выглядят сужения M на все указанного вида надмножества множества U, то мы знаем, как выглядит сужение M на U с точностью до перебора деревьев на компонентах связности.

Наконец, мы частично упорядочили множество клик по отношению включения, построив граф клик[6]. Тогда мы знаем для произвольной клики Cliquey, какие подклики лежат в ней. Значит, если мы знаем, как выглядит сужение минимального графа смежности M на каждую такую подклику, то можем понять, как выглядит сужение M на Clique j. Таким образом, поднимаясь по графу клик (а именно, перебирая его вершины таким образом, что на каждом шаге обрабатывается вершина графа клик, у которой либо нет вершин-сыновей, либо все ее вершины-сыновья уже обработаны) мы можем последовательно перебрать все минимальные графы смежности.

Определение 5.14. В ненаправленном дереве вершина V называется *листом* (или *висячей вершиной),* если из нее выходит ровно одно ребро.

Алгоритм Прюфера для перебора всех деревьев на n вершинах (1918) [12]. Цель — взаимооднозначо сопоставить деревьям последовательности из $n-2$ чисел[7], каждое из которых целое, не меньше нуля и не больше $n-1$.

Пусть изначально нам дано дерево. Его вершины пронумерованы числами от 0 до $n-1$. Мы хотим сопоставить этому дереву последовательность указанного выше вида, причем так, чтобы для разных деревьев (определенных, разумеется, на одном и том же наборе вершин) последовательности чисел были разными.

Будем это делать при помощи последовательных шагов.

На первом шаге мы находим вершину-лист с наименьшим номером, затем определяем вершину, с которой это лист соединен, и записываем ее номер в самую левую клетку последовательности номеров вершин (последовательность записывается слева направо). После чего найденный лист с минимальным номером исключается из дерева. (Ясно, что после исключения любой вершины-листа граф останется деревом, в частности, потому что, во-первых, он будет связным, так как через лист проходят простые пути лишь с концом в самом листе, во-вторых, в графе, в котором не было циклов, при исключении вершины циклы не появятся). На следующих шагах будет происходить почти то же самое.

На k-м шаге мы рассматриваем то дерево, которое осталось после выполнения предыдущих шагов. Находим в нем лист с наименьшим номером, берем номер вершины, с которой он соединен, и записываем этот номер в самую левую свободную клетку последовательности. После чего исключаем из дерева рассмотренный лист и переходим к следующему шагу.

Алгоритм прекращает свою работу, когда в дереве останется всего 2 вершины. Нетрудно заметить, что тогда в последовательности записано $n-2$ номера, так как в исходном дереве было $n-1$ ребро, а в получившемся — одно. При этом на каждом шаге мы исключали ровно по одному ребру.

Таким образом, мы построили по дереву числовую последовательность.

[6] Позднее будет показано, что в граф клик достаточно поместить только те клики, которые получаются перебором сужений максимального графа смежности по весам его ребер.

[7] Заметим, что такую последовательность чисел можно рассмотреть как последовательность цифр из записи числа в n-ичной системе счисления, т.е. собственно как n-ичное число.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Пусть теперь нам дана последовательность из $n-2$ чисел от 0 до $n-1$. Построим по ней дерево. Для этого заведем множество LT для записи специальных вершин.

На первом шаге мы возьмем крайнее слева число последовательности. Как мы помним по построению из предыдущей части описания алгоритма, к вершине с номером, равным выбранному числу, должен был крепиться лист с наименьшим номером. Подметим также, что номер листа в последовательности точно не встречается, так как его на первом же шаге исключили из дерева. Значит, нам надо соединить вершину, взятую из последовательности, с вершиной с наименьшим номером, не встречающейся в последовательности. После чего записать эту вершину, которая играет роль листа с минимальным номером, во множество LT. Теперь понятно, что множество LT будет представлять собой множество когда-то исключенных из рассмотрения при выполнении первого алгоритма вершин. Наконец, из последовательности исключается самое левое число (то, которое мы обработали) и алгоритм переходит к следующему шагу.

На k-м шаге алгоритм берет самую левую вершину из оставшейся последовательности. После чего ищет минимальную (по номеру) вершину из тех, которые, во-первых, не записаны в остающейся части последовательности, во-вторых, не лежат в LT. Она соединяется ребром с вершиной, взятой из последовательности, а также записывается во множество LT. Далее из последовательности вычеркивается самое левое (только-что рассмотренное нами) число. Алгоритм переходит к следующему шагу.

Шаги выполняются, пока в последовательности остались хоть какие-то числа, то есть в нем $n-2$ шага. Однако в дереве должно быть $n-1$ ребро. Чтобы провести последнее из них, вспомним, что после выполнения первого алгоритма в дереве остались две вершины, соединенные ребром. Причем понятно, что первый алгоритм их никогда не исключал. Следовательно, их нет во множестве LT. Заметим, что во множестве LT нет ровно двух вершин, так как на каждом шаге туда записывалось по одной вершине, а шагов — $n-2$. Следовательно, последнее ребро надо провести между теми двумя вершинами, которых нет в L_T. Дерево построено.

Теперь осталось заметить, что для перебора всех деревьев на n вершинах нам достаточно перебрать все последовательности указанного вида и по каждой построить дерево. Заметим, что таких последовательностей ровно $n^{(n-2)}$, так как в каждой клетке последовательности может стоять любое из n произвольных целых чисел от 0 до $n-1$. Значит, надо перебрать все числа от 0 до $(n^{(n-2)}-1)$, и каждому такому числу t сопоставить свою последовательность по следующему принципу: в k-ой клетке последовательности записана k-я цифра числа t в n-ичной системе счисления.

Формирование графа клик. Заметим, что в качестве весов клик достаточно перебирать веса ребер графа G_{max}. Объясним, почему это так.

Действительно, пусть $\mathrm{Clique}\,J$ — некоторая клика, такая, что ее вес (то есть пересечение весов ее вершин) не совпадает ни с одним из весов ее ребер. Такая клика априори может существовать и, формально, $\mathrm{Clique}\,J$ должна принадлежать графу клик. Однако мы не будем включать ее в граф клик, поскольку при исключении из Clique, ребер с весом $W(\mathrm{Clique},)$ получится одна компо-

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

нента связности (так как ребер с весом, равным весу W (Clique$j$), просто нет). Значит, Clique$j$ можно не рассматривать, так как все ребра Clicque$j$ описываются меньшими кликами.

После этого замечания граф клик построить несложно. Требуется перебрать все веса ребер максимального графа смежности G_{max} и посмотреть, какие клики на этих весах получаются. Это будут все клики, кроме исключенных выше в замечании. Ребра же между кликами проводятся согласно алгоритму Прюфера.

Переупорядочивание графа клик. Сопоставим каждой вершине графа клик число атомов в ее весе. Отсортируем вершины по сопоставленным им числам по убыванию (чем больше число — тем раньше вершина в списке).

Алгоритм построения минимальных графов смежности. В начале алгоритма дан набор весов **V** — т.е. некоторый набор подмножеств алфавита A. Можно считать, что он записан как массив натуральных чисел, где двоичная запись каждого натурального числа играет роль характеристического вектора соответствующего подмножества.

В первую очередь над заданным множеством вершин **V** строится максимальный граф смежности G_{max}. Далее в указанном графе смежности выделяются клики, которые формируют множество **Clique**. Напомним, что каждая клика Clique$j$ однозначно задается непустым множеством атомов Uj — Clique$j$ = G_{max} i Uj. Затем на множестве **Clique** строится граф клик C, кроме того, клики переупорядочиваются, как описано выше. Наконец, каждой клике Clique$j$ сопоставляется множество вершин Sj, которые в нее попали. Набор таких множеств обозначим **S**.

После построения базовых объектов создается первоначально пустой массив графов VC, которые можно образовать над вершинами **V**. Затем в массив VC записывается граф с пустым множеством ребер и набором вершин **V**.

Предполагается, что впоследствии в массиве VC будут формироваться графы смежности, построенные на множестве вершин **V**.

Дальнейший процесс основан на переборе элементов переупорядоченного набора клик **Clique**. Напомним, что перебор идет от клик с наибольшими весами к кликам с наименьшими весами: таким образом, выбрав очередную клику, мы уверены, что все ее подклики (если они существовали) были уже обработаны — веса подклик больше, чем вес самой клики, как это было отмечено ранее.

Схема алгоритма состоит из последовательности вложенных друг в друга переборов:

- перебираются клики согласно введенному на них упорядочиванию; каждая клика позволяет разбить на компоненты связности уже построенные объединения сужений минимальных графов на веса уже рассмотренных клик;
- перебираются с помощью описанного выше алгоритма Прюфера деревья, где вершинами являются выделенные компоненты связности, а ребра имеют вес, совпадающий с весом текущей клики, которые можно построить на компонентах связности;
- перебираются для каждого дерева на компонентах связности его возможные реализации на вершинах **V** (получившимися компонен-

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

тами связности P и Q ребро с весом клики Clicque$J$ можно провести IP x|Q| разными способами, т.е. таким ребром можно соединить любую вершину из P с любой вершиной из Q);

- перебираются все графы, накопившиеся в массиве VC, с каждым объединяется текущая реализация ребер (дерева клик) на вершинах V, каждый новый граф записывается в массив VC1.

После того, как одна клика обработана, VC чистится, туда копируется содержимое VC1, затем чистится сам VC1, чтобы принимать «пополненные» ребрами графы на следующих итерациях. Наконец, осуществляется переход к новой клике.

Рассмотрим некоторые шаги алгоритма, приведенные в его схеме выше, более пристально.

На J-том шаге алгоритм выбирает J-тую по переупорядочиванию клику Clique J , а также все ее подклики (т.е. всех сыновей клики в графе клик C). Впоследствии нам потребуется рассматривать сужения графов на подклики-сыновья.

Из массива VC берем граф H, хранящийся в VC[0]. Он будет нам нужен лишь для того, чтоб понять, какие же компоненты связности получатся при строгом сужении на Clique J . Заметим, что для выявления компонент связности можно брать произвольный граф из V, не обязательно хранящийся в нулевом элементе массива. Однако для определенности будем рассматривать именно тот, который выбрали.

Как мы ранее показали, для этого можно брать любой из минимальных графов. Тут стоит заметить, что все уже построенные графы суть объединение сужений минимальных графов на уже перебранные подклики. А так как все подклики Clique J уже перебраны, строгое сужение H (а лишь от него зависит множество компонент связности) на вес Clique J будет совпадать строгим сужением на вес Clique J любого покрывающего H минимального графа смежности. При этом стоит заметить, что хотя бы одни такой граф M существует, так как его можно получить, если зафиксировать все уже построенные ребра, а остальные выкидывать до тех пор, пока будет сохраняться условие смежности. Полученный граф будет минимальным, так как из него нельзя удалить никаких ребер без потери условия смежности. Это легко пояснить. Действительно, незафиксированные ребра мы не можем удалить просто по построению. Пусть есть такое ребро e, что $M \setminus e$ — тоже граф смежности. Так как $e \in H$, существует рассмотренная подклика Q, при обработке которой и добавлялось e. Но тогда сужение M на Q содержит (возможно, собственным образом) сужение H на Q, так как M содержит H . Но по построению H его сужение на любую уже рассмотренную подклику — минимальный граф смежности, то есть из этого сужения нельзя исключить ребро e, не потеряв условия смежности. Это противоречит нашему предположению о том, что e можно исключить. Следовательно, H достраивается хотя бы до одного минимального графа смежности. Значит, для определения компонент связности строго сужения H на вес Clique J можно брать минимальный граф смежности M , накрывающий H . А это — то же самое, что брать и само H, так как сужения на любую подклику Clique j у H и M

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

совпадают как по набору вершин, так и по набору ребер[8], и строгое сужение на $\mathrm{Clique}\,j$ есть объединение сужений на несобственные подклики.

В получившемся сужении графа H, хранившегося в массиве VC, алгоритм строит (более точно, выявляет) компоненты связности.

После того как множество выявленных компонент связности построено, на нем требуется перебрать все деревья; напомним, что вес ребер у этих деревьев будет равен весу клики, на которую производилось сужение в начале текущего шага. Деревья, как уже отмечалось, перебираются с помощью алгоритма Прюфера [12].

Пример 5.1. (Построение одного минимального графа смежности)

Пусть нам задан следующий набор весов вершин **V**: 0000.0011, 0000.0101, 0000.0111, 0010.1001, 0001.1001, 1000.1001.

Выполним шаги алгоритма построения графа смежности на этом наборе вершин. В первую очередь, построим G_{max} (рис. 5.1).

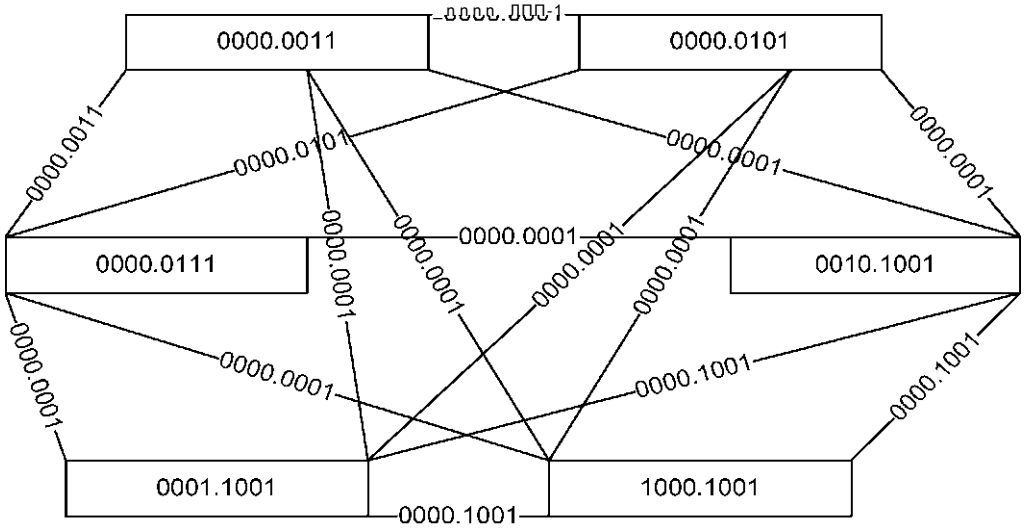
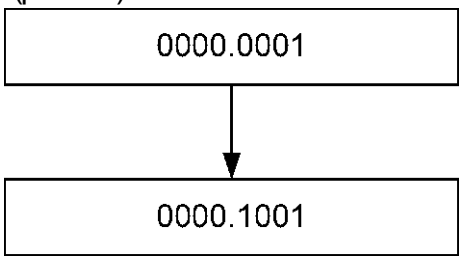


Рис. 5.1. Максимальный граф смежности для Примера 5.1.

Далее строим граф клик (рис. 5.2).



. 5.2. 5.1.

[8] Это утверждение требует некоторого пояснения. На самом деле, заметим, что сужение H на эту подклику содержится в сужении M на нее же. Пусть тогда сужение M содержит некоторое ребро e, которого нет в сужении H. Но такое невозможно ввиду минимальности M, так как его концы соединены некоторым путем по ребрам H, полностью лежащим в этой подклике, по построению H. Значит, сужения H и M на любую несобственную подклику $\mathrm{Clique}\,j$ совпадают как графы.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

После этого рассмотрим сужение на 0000.1001. (рис. 5.3)

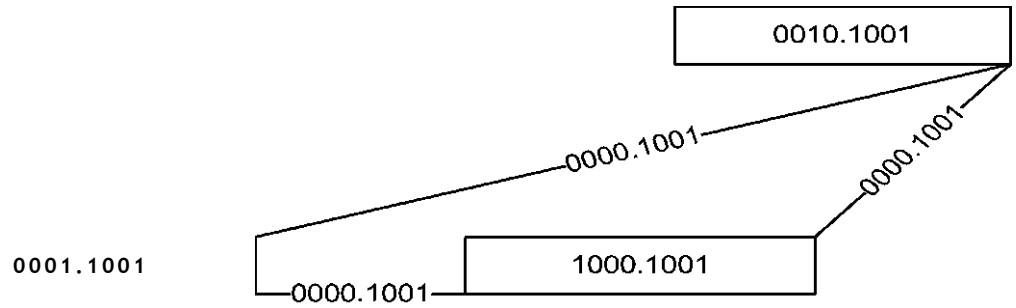


Рис. 5.3. Сужение максимального графа смежности на 0000.1001 для Примера 5.1.

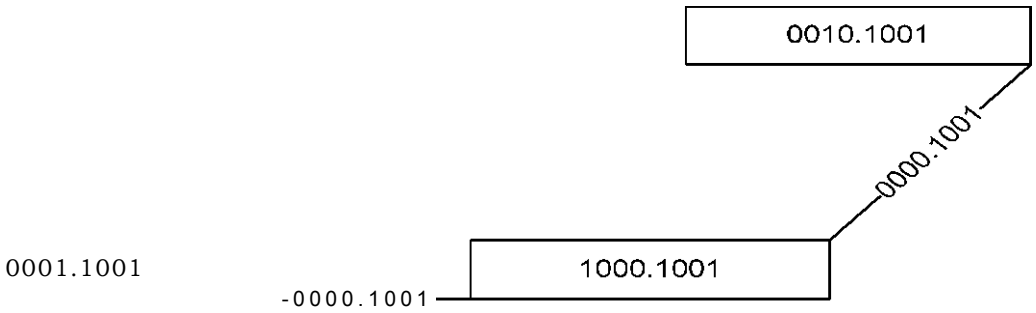


Рис. 5.4. Строгое сужение максимального графа смежности на 0000.1001 для Примера 5.1.

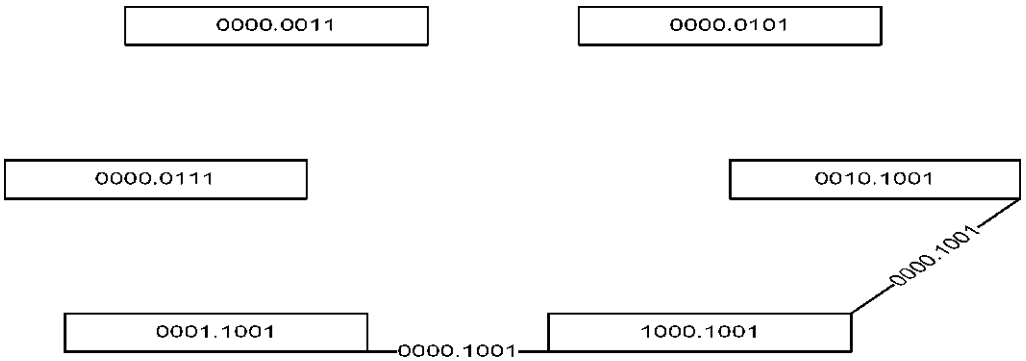


Рис. 5.5. Строгое сужение максимального графа смежности на 0000.0001 для Примера 5.1.

Затем построим все деревья на этом сужении (предварительно перейдя к строгому сужению, то есть, в данном примере, выкинув все ребра). Один из получившихся графов представлен на рис. 5.4.

Следующим шагом будет рассмотрение строгого сужения полученного графа на 0000.0001 (рис. 5.5).

На рис. 5.5. получились четыре компоненты связности. Теперь можно перебрать все деревья, которые можно построить на указанных компонентах связности, а также реализацию их ребер в графе смежности. Пример одного из получившихся графов смежности приведен на рис. 5.6.

Пример 5.2. (Построение нескольких минимальных графов смежности)

Пусть нам задан следующий набор вершин **V**: 001.000.011, 001.000.101, 001.001.001, 001.011.000, 001.101.000, 011.000.000, 101.000.000;

Выполним шаги алгоритма построения графа смежности на этом наборе вершин. В первую очередь, построим G_{max} (рис. 5.7).

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

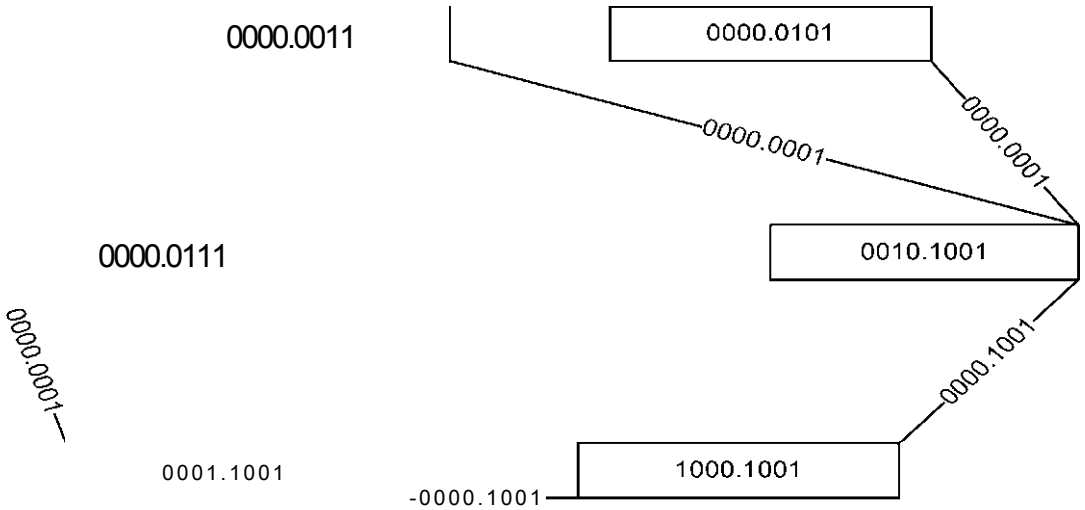


Рис. 5.6. Один из получившихся минимальных графов смежности в Примере 5.1.

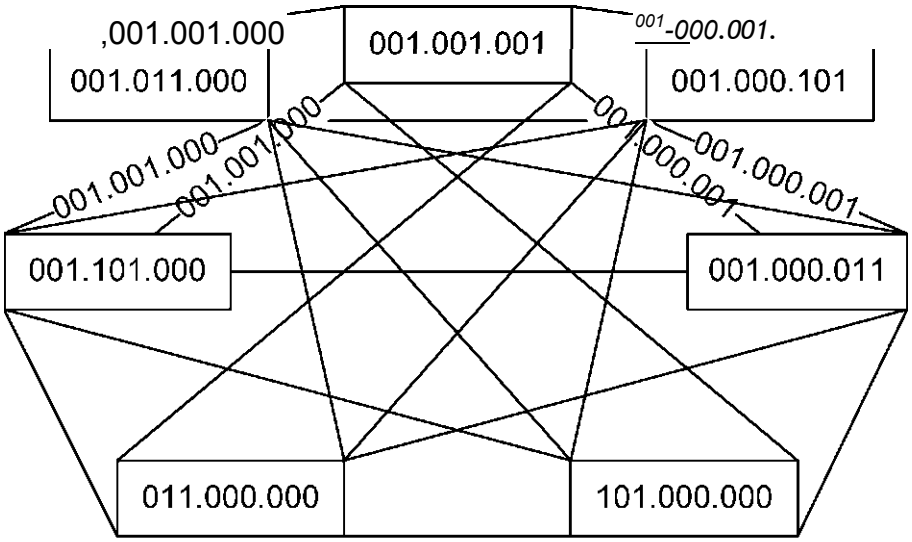


Рис. 5.7. Максимальный граф смежности для Примера 5.2.
Все ребра без пометок — с весом 001.000.000.

Далее строим граф клик (рис. 5.8).

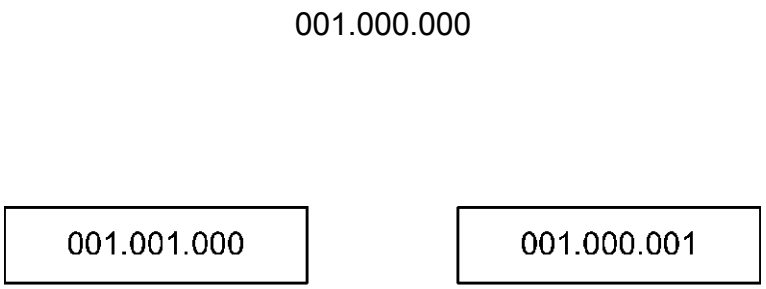


Рис. 5.8. Граф клик для Примера 5.2.

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

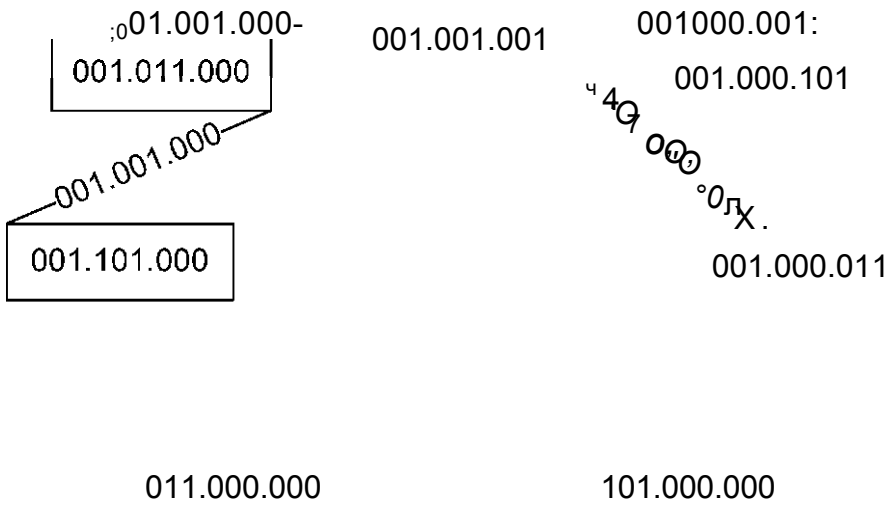


Рис. 5.8. Граф после рассмотрения весов 001.000.001 и 001.001.000 для Примера 5.2.

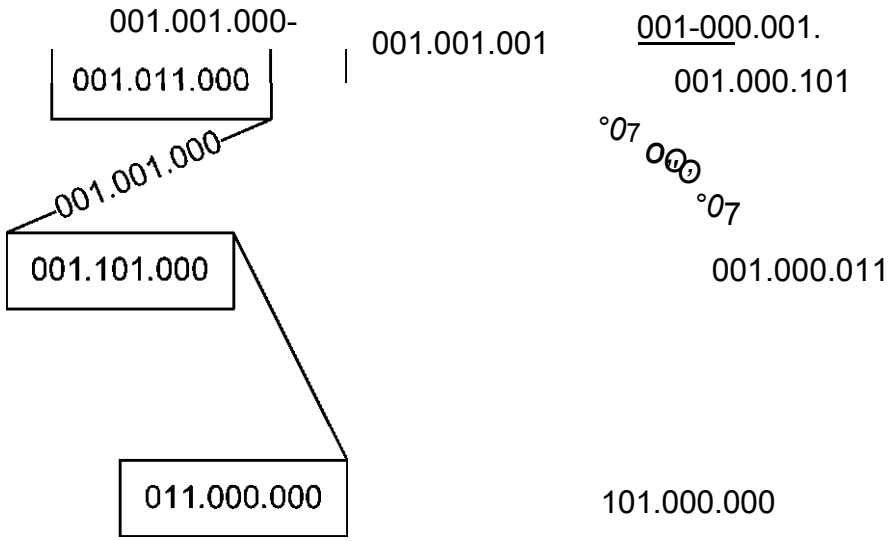


Рис. 5.9. Минимальный граф смежности (1) для Примера 5.2.

После этого, следуя алгоритму, надо рассмотреть сужение графа на 001.000.001 и построить в нем все деревья. Для краткости мы ограничимся одним деревом. Аналогично и с весом 001.001.000. В итоге, после этих двух шагов получится граф на рис. 5.9.

Затем, надо рассмотреть сужение на вес 001.000.000. В нем уже будут все вершины из исходного множества. Теперь надо выделить компоненты связности — их получится 3. В частности, наш пример иллюстрирует тот факт, что они не всегда совпадают с подкликами. По алгоритму надо построить все деревья на получившихся компонентах. Мы, чтобы остаться краткими, ограничимся в примере одним, например, таким, в котором компонента из пяти вершин соединена ребром с 011.000.000 и есть ребро между 011.000.000 и 101.000.000. У этого дерева на компонентах связности имеется много реализаций (их примеры — на рис. 5.9-5.11).

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

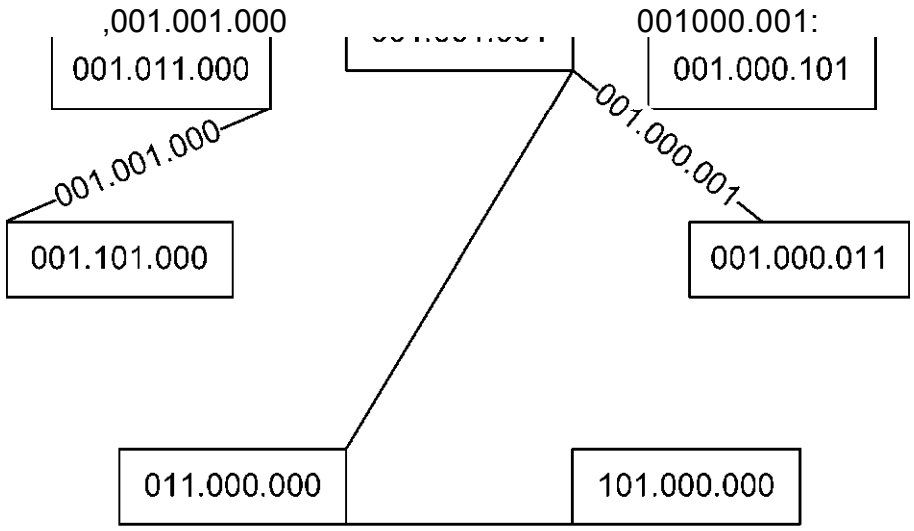


Рис. 5.10. Минимальный граф смежности (2) для Примера 5.2.

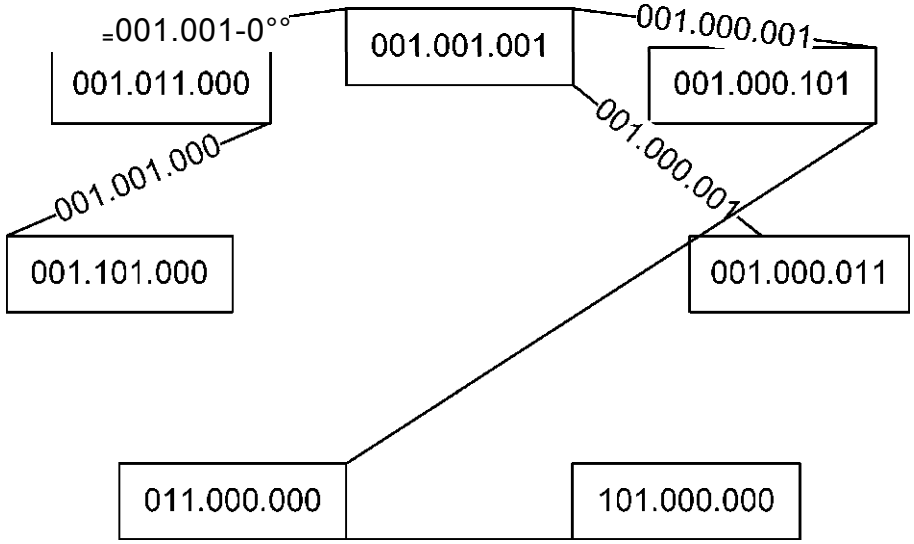


Рис. 5.11. Минимальный граф смежности (3) для Примера 5.2.

Это и есть минимальные графы смежности, но, конечно, не все. Например, тот, что изображен на рис. 5.8 при полном переборе по нашему алгоритму будет носить порядковый номер 239 в массиве VC. Всего же их будет 315.

6. Заключение

В настоящей работе были решены две задачи: 1) задача представления локальной структуры алгебраической байесовской сети с помощью системы классов и интерфейсов на языке Java и 2) задача построения глобальной структуры алгебраической байесовской сети в виде минимального дерева смежности.

В первой задаче рассматриваются фрагменты знаний, которые могут быть построены над набором квантов, либо идеалов дизъюнктов, либо идеалов

Труды СПИИРАН. 2007. Вып. 5. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 5. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

конъюнктов. Фрагменты знаний также различаются видами оценок: интервальные, скалярные, бинарные, — что, в свою очередь, послужило причиной построения набора различных интерфейсов, отвечающих за функциональность, связанную с чтением и записью данных из/в ФЗ. Фрагменты знаний также могут играть роль свидетельств: неопределенного, стохастического бинарного. В зависимости от доступных исходных данных для априорного или апостериорного вывода используются система интерфейсов и классов, представляющих семейства машин вывода Inferrer (априорный вывод) и Propagator (апостериорный вывод).

Решение второй задачи состоит в том, что удалось предложить алгоритм построения и перебора всех минимальных деревьев смежности, которые можно построить над заданным набором фрагментов знаний.

Литература

1. *Абрамян А. К.* Комплекс программ для локального логико-вероятностного вывода в алгебраических байесовских сетях: разработка прототипа ядра (Java- и ILOG-технологии). Дипломная записка / Под. рук. А. Л. Тулупьева. СПб., 2007. 262 с. (Санкт-Петербургский государственный . - . . .)
2. *Беллман Р.* Введение в теорию матриц. М.: Наука, Гл. редакция физико-математической литературы, 1969. 368 с.
3. *Городецкий В. И., Тулупьев А. Л.* Формирование непротиворечивых баз знаний с неопределенностью // Изв. РАН. Сер. Теория и системы управления. 1997. Т. 5. С. 33-42.
4. *Городецкий В. И.* Алгебраические байесовские сети — новая парадигма экспертных систем // Юбилейный сборник трудов институтов отделения информатики, вычислительной техники и автоматизации РАН. Т. 2. М.: РАН, 1993. С. 120-141.
5. *Тулупьев А. Л.* Алгебраические байесовские сети: локальный логико-вероятностный вывод: Учеб. пособие. СПб.: СПбГУ; ООО Издательство «Анатолия», 2007. 80 с. (Сер. Элементы мягких вычислений).
6. *Тулупье в 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).
7. *Тулупье в А. Л., Николенко С. И., Сироткин А. В.* Байесов ские сети: логико-вероятностный подход. СПб.: Наука, 2006. 607 с.
8. *Тулупье в А. Л.* Ациклические алгебраи ческие байесовские сети: логико-вероятностный вывод // Нечеткие системы и мягкие вычисления: Научный журнал Российской ассоциации нечетких систем и мягких вычислений. 2006. Том 1, № 1. С. 57-93.
9. *Тулупье в А. Л.* Генерация множества ограничений на распределение оценок вероятности над идеалом цепочек конъюнкций // Вестник молодых учёных. 2004. № 4. Серия: Прикладная математика и механика. 2004. № 1. С. 35-43.
10. *Тулупье в А.Л., Абрамян АХ.* Ло гико-вероятностный вывод в направленном БСД-цикле // Труды СПИИРАН. 2007. Вып. 4. СПб.: Наука, 2007. С. 87-118.
11. *Тулупье в А. Л. и Никитин Д. А.* Экстремальные задачи в апостериорном выводе над идеалами цепочек конъюнкций // Труды СПИИРАН. 2005. Вып. 2, т. 2. СПб.: Наука, 2005. С. 12-52.
12. *Bondy J. A., Murty J. S. R.* Graph Theory with Applications. New York: Elsevier Science Publishing Co., 1976. 264 p.
13. *Fagin R., Halpern J. Y., Megiddo N.* A Logic for Reasoning about Probabilities. Report RJ 6190 (60900) 4/12/88. pp. 1-41.
14. *Fagin R., Halpern J. Y.* Uncertainty, Belief, and Probability-2 // Proc. of the IEEE Symposium on Logic and Computer Science. 1991. Vol. 7. P. 160-173.
15. *Nilsson N. J.* Probabilistic Logic // Artificial Intelligence. 1986. Vol. 47. Amsterdam: Elsevier Science Publishers B.V., 1986. P. 71-87.
16. *Nilsson N. J.* Probabilistic Logic Revisited // Artificial Intelligence. 1993. Vol. 59. Amsterdam: Elsevier Science Publishers B.V., 1993. P. 31-36.