УДК 004.825

С. Д. Махортов, д-р физ.-мат. наук, доц., зав. каф., e-mail: sd@expert.vrn.ru,
А. Н. Шмарин, аспирант, Воронежский государственный университет

Нечеткий LP-вывод и его программная реализация

Предложена модификация LP-вывода, базирующаяся на идее динамического программирования — разбиении основной задачи на отдельные части (подзадачи) и решении каждой подзадачи только один раз. Приведены особенности "быстрого" расчета степеней истинности решений. Представлен способ улучшения качества решений в кластерно-релевантном LP-выводе, основанный на вычислении оценок возможных решений без построения прообразов. Описана программная реализация.

***Ключевые слова**: LP-вывод, И/ИЛИ-граф, динамическое программирование, нечеткие правила, программная реализация*

Введение

Теория LP-структур предоставляет эффективный аппарат для исследования процессов управления знаниями, верификации знаний, оптимизации логического вывода в различных разделах информатики [1—3]. В частности, в статье [1] представлены усовершенствованные алгоритмы обратного вывода, основанные на решении продукционно-логических уравнений в LP-структурах. Стратегия релевантного LP-вывода направлена на минимизацию числа обращений к внешним источникам информации.

В настоящей работе описана новая идея оптимизации LP-вывода. Ее преимущество обеспечивается построением дополнительной структуры данных для хранения найденных результатов, что позволяет экономить время при обходе графа базы знаний. Такой структурой является дерево достижимости ИЛИ-вершин, хранящее в своих узлах информацию для построения всех прообразов объекта экспертизы [1]. Еще одна особенность полученных в ходе настоящего исследования результатов — возможность обработки нечетких правил. Кроме того, впервые описана компьютерная реализация предложенных идей. Снижение сложности обхода достигается в силу того обстоятельства, что достаточно лишь один раз исследовать поддеревья базы знаний с корнем в вершине, выводимой более чем одним правилом, и использовать полученную информацию при обходе других связанных вершин. Таким образом, LP-вывод можно выполнять в два этапа:

- построение дерева достижимости ИЛИ-вершин;
- обход созданного дерева достижимости и построение прообразов.

Одним из важных результатов общей теории LP-структур является тот факт, что частное решение продукционно-логического уравнения на решетке существует тогда и только тогда, когда соответствующий слою отношения ориентированный граф не содержит циклов [1]. В силу данного обстоятельства, для корректной работы описываемого ниже метода необходимо исключить из рассмотрения все слои с циклами. Однако такое ограничение нетривиально и его практическая реализация требует отдельного исследования. Далее в работе будет подразумеваться, что все слои с циклами заранее исключены из рассмотрения.

1. Базовые определения

Используемая в настоящей работе терминология соответствует публикациям [1—5]. Введем несколько дополнительных понятий.

Утверждение об объекте базы знаний является *ИЛИ-вершиной*, если оно выводится более чем одним правилом.

Если более чем одно правило выводит одно и то же утверждение об объекте базы знаний, то будем считать, что такие правила *выводят ИЛИ-вершину*. Каждое такое правило будем называть *ИЛИ-связью*.

Деревом вывода утверждения называется поддерево базы знаний, корнем которого является данное утверждение.

Утверждение *участвует в выводе предпосылки правила*, если оно включено в предпосылку, либо включено в дерево вывода одного из утверждений предпосылки.

ИЛИ-вершина является *непосредственно достижимой* из ИЛИ-связи, если данная вершина участвует в выводе ИЛИ-связи и между ними существует путь, не включающий в себя другие ИЛИ-вершины. Такая ИЛИ-вершина будет являться *потомком* ИЛИ-связи. Соответственно ИЛИ-связь будет называться *предком* ИЛИ-вершины.

2. Построение дерева достижимости ИЛИ-вершин

Для создания дерева достижимости необходимо, чтобы вершины графа базы знаний удовлетворяли перечисленным ниже требованиям.

1. Каждой ИЛИ-связи должны быть поставлены в соответствие следующие структуры данных:

- LP-код [5] — для хранения информации о начальных утверждениях, участвующих в выводе предпосылок правила;
- список потомков — для хранения индексов ИЛИ-вершин, являющихся прямыми потомками.

2. Каждая ИЛИ-вершина должна иметь список предков — для хранения индексов ИЛИ-связей, являющихся прямыми предками.

3. Ребра должны создаваться между ИЛИ-связью и ее потомками — ИЛИ-вершинами.

Узлами дерева достижимости будут являться ИЛИ-вершины и выводящие их правила, а ребрами — отношения "предок/потомок". Корни дерева — это вершины, соответствующие целевым утверждениям. Ребрам, исходящим из корней, соответствуют правила, выводящие эти утверждения. Создание ребра между ИЛИ-связью и ИЛИ-вершиной выполняется следующим образом:

- в список потомков ИЛИ-связи добавляется индекс ИЛИ-вершины;
- в список предков ИЛИ-вершины добавляется индекс ИЛИ-связи.

При построении дерева достижимости необходимо использовать общий указатель для хранения индекса текущей ИЛИ-связи. Будем называть этот указатель *маркером верхней ИЛИ-связи*.

Алгоритм построения дерева достижимости ИЛИ-вершин

1. Выбрать очередное целевое утверждение.

2. Обойти все правила, выводящие текущее утверждение.

2.1. При выборе очередного правила сохранить предыдущее значение маркера верхней ИЛИ-связи и присвоить ему индекс выбранного правила.

2.2. Выполнить обход утверждений, входящих в предпосылку правила, и их деревьев вывода.

2.3. Если на очередном шаге обхода встретилось утверждение, выводящееся более чем одним правилом — создать в дереве достижимости ребро между данным утверждением и правилом, на которое указывает маркер верхней ИЛИ-связи. Если данное утверждение имеет статус "не открыто", присвоить ему статус "открыто" и выполнить для него Шаг 2.

2.4. Если достигнуто начальное утверждение, то его необходимо добавить к LP-коду правила, на которое указывает маркер верхней ИЛИ-связи.

2.5. Когда завершен обход предпосылок текущей ИЛИ-связи, необходимо восстановить сохраненное ранее значение маркера верхней ИЛИ-связи.

Очевидно, что вычислительная сложность построения дерева достижимости равна сложности обхода графа в глубину, т. е. $O(N)$, где N — число правил в базе знаний. Память, занимаемая деревом достижимости, не превышает объем памяти для графа базы знаний.

3. Вычисление степеней истинности

При обходе дерева нечеткой базы знаний не всегда можно однозначно рассчитать степень истинности утверждений [6]. Причина в том, что в дереве вывода могут встречаться ИЛИ-узлы. Правила, выводящие данные узлы, могут становиться невыполнимыми при конкретизации начальных объектов.

Вариантом решения отмеченной задачи может служить вычисление степеней истинности после построения прообразов. Однако такой подход приводит к увеличению вычислительной сложности и отсутствию возможности реализации кластерно-релевантного LP-вывода [1]. Более продуктивной оказывается реализация степеней истинности формулами, содержащими неизвестные переменные. Такие формулы могут быть представлены в виде дерева, узлами которого являются операции T-нормы и S-нормы [6].

Пусть сформулировано некоторое утверждение об объекте базы знаний, выводимое лишь одним правилом. Если все факты в предпосылке этого правила являются начальными, либо все соответствующие им деревья вывода не включают ИЛИ-вершин, то степень истинности исходного утверждения может быть

рассчитана в виде числового значения. Такую степень истинности будем называть *заданной явно.*

Дерево, моделирующее формулу расчета степеней истинности утверждения, будем называть *деревом истинности вершины.*

Замечание 1. Для корректности описываемого подхода необходимо, чтобы операция Т-нормы удовлетворяла условию $T(x, 0) = 0$. Данное условие выполняется для большинства применяемых на практике классов Т-норм.

Структура дерева истинности вершины. Каждый узел дерева соответствует структуре данных, которая включает:

- идентификатор типа операции, принимающий значения "Т-норма" либо "S-норма";
- числовое значение степени истинности;
- список аргументов, каждый элемент которого представляет ссылку на дерево истинности вершины, соответствующей аргументу.

Если узел моделирует Т-норму, то после заполнения списка аргументы с явно заданной степенью истинности необходимо удалить. Вместо них создается новый аргумент, для которого степень истинности равна результату применения операции Т-нормы к степеням истинности удаленных узлов. Данное преобразование возможно в силу того, что Т-норма коммутативна и ассоциативна по определению.

В худшем случае узел дерева истинности может иметь список аргументов, длина которого близка к числу вершин M графа базы знаний. Другими словами, верхняя оценка числа аргументов узла равна $O(M)$. При этом каждый узел дерева истинности в свою очередь ассоциирован с соответствующей вершиной графа базы знаний. Поэтому дерево истинности имеет верхнюю оценку объема используемой памяти $O(M^2)$. Поскольку построение дерева истинности заключается в последовательном построении всех его узлов, то оценка вычислительной сложности такого действия также равна $O(M^2)$.

Замечание 2. Если степень истинности задана явно, то дерево вырождается в узел, хранящий данное числовое значение.

Замечание 3. Если задано числовое значение степени истинности узла, то список аргументов узла считается пустым.

Описанный способ сокращения аргументов Т-нормы нельзя применять для операции S-нормы. Данное ограничение связано с тем, что моделирующие S-норму узлы создаются для ИЛИ-вершин, на основе которых могут быть получены различные прообразы.

Сохранение структурного представления степеней истинности необходимо для того, чтобы на этапе построения прообразов была возможность быстро (без обхода графа базы знаний) выполнить следующие действия:

- расчет степени истинности целевого утверждения для каждого прообраза;
- проверку условия, можно ли увеличить степень истинности выведенного целевого утверждения с помощью конкретизации начального объекта базы знаний;
- при выполнении кластерно-релевантного LP-вывода [1] — расчет оценки максимально возможной степени истинности для подграфа базы знаний, определяющего группу прообразов, и использование данного показателя для принятия решения о необходимости построения соответствующих прообразов.

Создание деревьев истинности вершин необходимо выполнять на этапе построения дерева достижимости. При этом кроме создания деревьев истинности для утверждений, необходимо строить деревья истинности для правил, выводящих ИЛИ-вершины.

4. Обход дерева достижимости ИЛИ-вершин и расчет показателей релевантности

Построение множества минимальных начальных прообразов позволяет выполнять начальный расчет и последующую модификацию коэффициентов, на основе которых выбирается наиболее подходящий для конкретизации (релевантный) объект экспертизы.

Основными коэффициентами (показателями релевантности) тестируемого объекта являются следующие параметры [1]:

- счетчик вхождений в прообразы;
- число вхождений в прообразы с малой мощностью;
- число вхождений в прообразы с высокой степенью истинности.

Однако построение всего множества минимальных начальных прообразов имеет экспоненциальную сложность и требует экспоненциального объема памяти для их хранения. Решение этой задачи заключается в том, чтобы не вычислять прообразы в явном виде. Это возможно в силу того обстоятельства, что слои с циклами исключены из рассмотрения. Дело в том, что дерево достижимости хранит всю необходимую информацию о прообразах, и каждый путь в нем соответствует слою. Так как слои с циклами исключены, оставшиеся слои всегда имеют решение. Для построения решения необходимо обойти соответствующий путь, объединить LP-коды и упростить дерево истинности до числового значения.

В силу того, что слой эквивалентен пути в дереве достижимости, следует ряд отмеченных далее важных фактов.

1. Задача подсчета числа вхождений каждого атома в прообразы может быть сведена к задаче вычисления числа всевозможных путей от листовых вершин дерева достижимости к корневым.

Из теории графов известно [7], что матрица смежности, возведенная в степень k, дает информацию обо всех путях длины k. Пусть $\mathbf{E}$ — матрица смежности дерева достижимости. Чтобы получить информацию о числе всевозможных путей длины от 1 до k между любыми двумя вершинами, достаточно вычислить суммарную матрицу $\mathbf{E} + \mathbf{E}^2 + \mathbf{E}^3 + ... + \mathbf{E}^k$. Самый длинный путь в графе без циклов не может превышать число вершин в графе, уменьшенное на единицу. Это значит, что для получения информации о всевозможных путях между любыми двумя вершинами в дереве достижимости, необходимо вычислить сумму $\mathbf{D} = \mathbf{E} + \mathbf{E}^2 + \mathbf{E}^3 + ... + \mathbf{E}^{S-1}$, где S — число вершин в дереве достижимости.

Простой алгоритм перемножения матриц имеет кубическую сложность, а матрица смежности имеет размер $S \times S$. Из приведенного выше следует, что оценка вычислительной сложности для задачи подсчета всевозможных путей в дереве достижимости равна $O(S^4)$, т. е. необходимо S раз найти произведение матрицы размером $S \times S$ на себя.

При подсчете всевозможных путей целесообразно вести стек LP-кодов, добавляя в него новый элемент при посещении очередной вершины. Новый LP-код должен вычисляться путем сложения (с контролем противоречивости) предыдущего элемента стека и LP-кода текущей ИЛИ-связи. Если при вычислении нового LP-кода возникло противоречие, необходимо завершить обход текущей ветви и вернуться назад.

Таким образом, общая оценка сложности этой задачи равна $O(S^4M_0)$, где M_0 — число начальных утверждений в базе знаний. Верхняя же оценка объема памяти стека LP-кодов будет равна $O(M_0S)$.

Замечание 4. При использовании быстрых алгоритмов перемножения матриц, асимптотическая оценка будет ниже.

2. Слои могут быть объединены в группы, соответствующие связным подграфам дерева достижимости. Нахождение прообразов минимальной мощности и максимальной степени истинности может быть выполнено для группы слоев. Таким образом, вычисление числа вхождений в прообразы с малой мощностью или высокой степенью истинности без построения самих прообразов можно выполнить следующим способом.

♦ В дереве достижимости для каждой ИЛИ-вершины необходимо определить правила (ИЛИ-связи), которые могли бы выводить данную вершину применением начальных утверждений, соответствующих атомам

— прообраза минимальной мощности;

— прообраза максимальной степени истинности.

Данное действие можно выполнить путем обхода дерева достижимости в глубину. С каждым правилом (ИЛИ-связью) при этом необходимо связать два значения:

- минимальную мощность прообраза, который мог бы быть найден для поддерева вывода предпосылок этого правила, если бы утверждение в заключении правила было целью;
- максимальную степень истинности, которая может быть получена для заключения правила.

♦ После выполнения обхода в глубину для целевых утверждений будут известны максимально возможная степень истинности и минимально возможная мощность прообраза, а также пути в графе, по которым можно достигнуть листовых вершин.

Для вычисления показателей релевантности будет достаточно выполнить обход от корня к листам по ребрам, для которых найдены наилучшие оценки (наибольшая степень истинности и наименьшая мощность). В процессе обхода необходимо вести стек LP-кодов и проверять элемент, добавляемый в стек, на противоречивость. Если обнаружено противоречие, обход текущей ветви завершается. Если же противоречий нет, нужно обновить показатели релевантности для объектов из LP-кода текущей ИЛИ-вершины.

Вычислительная сложность выполняемого обхода зависит от числа путей с наилучшими оценками. В худшем случае это будут почти все возможные пути в графе. Однако такое возможно лишь для реберно M-связных графов баз знаний, либо для графов, реберная степень связности которых близка к числу M (т. е. к числу вершин в графе базы знаний). Если же степень реберной связности графа базы знаний близка к единице, то число всевозможных путей с наилучшими оценками будет в худшем случае полиномиальным [7].

Таким образом, можно сделать следующие выводы о вычислительной сложности описанного метода.

- Если граф базы знаний имеет степень реберной связности, близкую к числу вершин в данном графе, то сложность описанного метода экспоненциальна.
- Если граф базы знаний имеет степень реберной связности, близкую к единице, то оценка вычислительной сложности описываемого метода будет равна $O(N) + O(M^2) + O(S^4M_0) + O(S^2P) = O(N + M^2 + S^4M_0 + S^2P)$, где N — число правил в базе знаний; M — число вершин в графе базы знаний; M_0 — число начальных утверждений в базе знаний; S — число вершин в дереве достижимости; P — число всевозможных путей в дереве достижимости, посещаемых при обходе ребер с наилучшими оценками.

5. Программная реализация

Описанный метод реализован в программной системе fLPExpert (fuzzy-LPExpert). Данный продукт является кроссплатформенным программным обеспечением, распространяемым по лицензии GPL 2.1 и работающим в операционных системах Windows и Linux. Получить последнюю версию исполняемых файлов, исходных кодов, дополнительную информацию о работе, архитектуре и методике развертывания системы можно в Интернете: http://sourceforge.net/projects/lpexpert.

Система реализована на языке C++ с использованием библиотеки Qt4. Описанный метод включен в систему в тестовом режиме и может быть активизирован через настройки проекта. На текущий момент полноценное практическое применение описанного метода представляет определенные трудности в силу того, что алгоритм корректного исключения из рассмотрения слоев с циклами еще не разработан.

Реализация описанных алгоритмов выполнена в классе LPGraph и состоит из следующих компонентов.

- Подготовка данных. Активизируется методом LPGraph::prepare(). Он удаляет из базы знаний правила, которые могут породить цикл. С точки зрения корректности системы в общем случае так поступать нецелесообразно. Данный подход применяется лишь для подготовки корректных тестовых данных.
- Построение дерева истинности и дерева достижимости. Запускается вызовом метода LPGraph::build(). Дерево достижимости представлено в виде класса ReachabilityThree, который представляет собой контейнер для списка смежности. Узлы дерева истинности являются экземплярами класса TruthNode и размещаются в контейнерах заключений правил — в элементах Trule::Perm. Данные функциональные возможности соответствуют действиям, описанным в разд. 2 и 3.
- Вычисление показателей релевантности и индекса объекта, наиболее подходящего для конкретизации. Активизируется методом LPGraph::getToAsk(). Он выполняет действия, описанные в разд. 4.
- Присваивание конкретного значения объекту базы знаний. Выполняется методом LPGraph::setValue (int object, const TValue & value). После конкретизации объекта выполняется прореживание пространства поиска путем обновления дерева достижимости и исключения из него всех подграфов, обход которых стал невозможен.
- Проверка наличия ответа. Реализуется методом LPGraph::hasAnswer(). Метод возвращает значение ответа либо 0, если информации для вывода ответа недостаточно. Результат −1 соответствует ответу "решений нет".

Заключение

В статье рассмотрены теоретические особенности оптимизированного алгоритма LP-вывода с поддержкой нечеткости на уровне коэффициентов доверия. Предложены алгоритмические решения для улучшения качества решений кластерно-релевантного LP-вывода. Получены оценки сложности рассматриваемых алгоритмов. Описана программная реализация представленных идей.

Эффективность и практическая значимость разрабатываемого подхода заключается в существенном снижении объема вычислений, что дает возможность применения метода LP-вывода на больших базах знаний, а также и в более сложных логических системах информатики [8].

Список литературы

1. **Болотова С. Ю., Махортов С. Д.** Алгоритмы релевантного обратного вывода, основанные на решении продукционно-логических уравнений // Искусственный интеллект и принятие решений. 2011. № 2. С. 40—50.
2. **Махортов С. Д.** LP-структуры для обоснования и автоматизации рефакторинга в объектно-ориентированном программировании // Программная инженерия. 2010. № 2. С. 15—21.
3. **Махортов С. Д.** Основанный на решетках подход к исследованию и оптимизации множества правил условной системы переписывания термов // Интеллектуальные системы. 2009. Т. 13. Вып 1—4. С. 51—68.
4. **Levi G., Sirovich F.** Generalized And/Or Graphs // Artificial Intelligence Journal. 1976. № 7. P. 243—259.
5. **Махортов С. Д.** Интегрированная среда логического программирования LPExpert // Информационные технологии. 2009. № 12. С. 65—66.
6. **Заде Л. А.** Понятие лингвистической переменной и его применение к принятию приближенных решений: пер. с англ. М.: Мир, 1976. 165 с.
7. **Харари Ф.** Теория графов: пер. с англ. М.: Едиториал УРСС, 2003. 296 с.
8. **Чечкин А. В.** Нейрокомпьютерная парадигма информатики // Нейрокомпьютеры: разработка, применение. 2011. № 7. С. 3—9.