УДК 004.825

ОПТИМИЗАЦИЯ МЕТОДА LP-ВЫВОДА

Махортов С.Д., Шмарин А.Н.

Воронежский государственный университет, г. Воронеж

Аннотация. Предложен способ оптимизации *LP*-вывода, основанный на идее динамического программирования – разбиении основной задачи на отдельные части (подзадачи) и решении каждой подзадачи только один раз. Возможность применения данного подхода обусловлена свойствами *ИЛИ*-вершин в *И/ИЛИ*-графе. Приведено описание особенностей «быстрого» расчета степеней истинности решений. Представлен способ улучшения качества решений в кластерно-релевантном LP-выводе, основанный на создании оценок возможных решений без построения прообразов.

Ключевые слова: LP-вывод, И/ИЛИ-граф, динамическое программирование, нечеткие правила.

🗁🖙 ВВЕДЕНИЕ

Теория LP-структур предоставляет эффективный алгебраический аппарат для исследования процессов управления знаниями, верификации знаний, оптимизации логического вывода в различных разделах информатики [1–2]. 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 данной работе описана новая идея оптимизации LP-вывода. Она заключается в построении дополнительной структуры данных для хранения уже найденных результатов, что позволяет экономить время при обходе графа базы знаний. Такой структурой данных является дерево достижимости *ИЛИ*-

вершин, хранящее в своих узлах информацию, необходимую для построения всех прообразов объекта экспертизы. Другая особенность настоящего исследования – возможность обработки нечетких правил.

Снижение сложности обхода достигается в силу того, 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 2 этапа:

1) построение дерева достижимости *ИЛИ*-вершин;
2) обход созданного дерева достижимости и построение прообразов.

Одним из фундаментальных результатов общей теории LP-структур является доказательство того факта, что частное решение продукционно-логического уравнения на решетке существует тогда и только тогда, когда соответствующий слою отношения ориентированный граф не содержит петель [1]. При этом общее решение вычисляется как объединение всех частных решений. В силу этого обстоятельства, для корректной работы описываемого ниже метода необходимо предварительно удалить из графа базы знаний ребра, порождающие слои с ориентированными циклами.

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Используемая в настоящей работе терминология соответствует публикациям [1–4]. Введем несколько дополнительных понятий.

Определение 1. Утверждение об объекте базы знаний является *ИЛИ-вершиной*, если оно выводится более чем одним правилом.

Определение 2. Если более чем одно правило выводит одно и то же утверждение об объекте базы знаний, то будем говорить, что такие правила *выводят ИЛИ-вершину*. Каждое такое правило будем называть *ИЛИ-связью*.

Определение 3. *Деревом вывода утверждения* называется поддерево базы знаний, корнем которого является данное утверждение.

Определение 4. Утверждение *участвует в выводе предпосылки правила*, если оно включено в предпосылку либо включено в дерево вывода одного из утверждений предпосылки.

Определение 5. *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.

ПОСТРОЕНИЕ ДЕРЕВА ДОСТИЖИМОСТИ *ИЛИ*-ВЕРШИН

Для создания дерева достижимости необходимо, чтобы вершины графа базы знаний удовлетворяли следующим требованиям:

1. Для всех правил и утверждений должны быть заданы индексы.
2. Каждой *ИЛИ*-связи должны быть сопоставлены следующие структуры данных:
    2.1. *LP*-код [4] – для хранения информации о начальных утверждениях, участвующих в выводе предпосылок правила.
    2.2. Список потомков – для хранения индексов *ИЛИ*-вершин, являющихся прямыми потомками.
3. Каждая *ИЛИ*-вершина должна иметь список предков – для хранения индексов *ИЛИ*-связей, являющихся прямыми предками.
4. 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:

1. В список потомков *ИЛИ*-связи добавляется индекс *ИЛИ*-вершины.
2. 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.

Вычислительная сложность построения дерева достижимости равна сложности обхода графа в глубину, то есть O(*N*), где *N* — количество правил в базе знаний. Память, занимаемая деревом достижимости не превышает объем памяти, занимаемой графом базы знаний.

ВЫЧИСЛЕНИЕ СТЕПЕНЕЙ ИСТИННОСТИ

При обходе дерева базы знаний не всегда можно рассчитать численное значение степени истинности [5] утверждений. Причина в том, что в дереве вывода могут встречаться *ИЛИ*-узлы. Правила, выводящие данные узлы, могут становиться невыполнимыми при конкретизации начальных объектов. Вариантом решения данной проблемы может служить вычисление степеней истинности после построения прообразов. Однако такой подход приводит к увеличению вычислительной сложности и невозможности реализации кластерно-релевантного LP-вывода [1].

Более продуктивным является реализация степеней истинности в виде формул, содержащих неизвестные переменные. Такие формулы могут быть представлены в виде дерева, узлами которого являются операции T-нормы и S-нормы [5].

Определение 6. Пусть задано некоторое утверждение об объекте базы знаний, выводимое лишь одним правилом. Если данное утверждение является начальным, либо дерево вывода утверждения не включает в себя *ИЛИ*-вершин, то степень истинности такого утверждения может быть рассчитана в виде численного значения. Такую степень истинности будем называть *заданной явно*.

Определение 7. Дерево, моделирующее формулу расчета степеней истинности утверждения, будем называть *деревом истинности вершины*.

Замечание 1. Для корректности описываемого подхода необходимо, чтобы операция T-нормы удовлетворяла следующему условию T(x, 0) = 0. Данное

условие выполняется для большинства применяемых на практике классов T-норм.

Структура дерева истинности вершины.

Каждый узел дерева соответствует структуре данных, которая включает в себя:

1. Идентификатор типа операции, принимающий значения «T-норма» либо «S-норма».
2. Числовое значение степени истинности.
3. Список аргументов, каждый элемент которого является ссылкой на дерево истинности вершины, соответствующей аргументу. Если узел моделирует T-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 T-нормы к степеням истинности удаленных узлов. Данное преобразование возможно в силу того, что T-норма коммутативна и ассоциативна по определению.

Объем отдельного узла дерева истинности имеет оценку O(M), где M — количество вершин в графе базы знаний. Объем всего дерева истинности равен O(M^2). Сложность построения дерева истинности также имеет оценку O(M^2).

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

Замечание 3. Если задано числовое значение степени истинности узла, то список аргументов узла будет пуст.

Описанный способ сокращения аргументов T-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 имелась возможность быстро (без обхода графа базы знаний) выполнить следующие действия:

1. Расчет степени истинности целевого утверждения для каждого прообраза.
2. Проверка условия, можно ли увеличить степень истинности выведенного целевого утверждения с помощью конкретизации начального объекта базы знаний.
3. 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 выполнять создание деревьев истинности для правил, которые выводят *ИЛИ*-вершины.

ОБХОД ДЕРЕВА ДОСТИЖИМОСТИ *ИЛИ*-ВЕРШИН И РАСЧЕТ ПОКАЗАТЕЛЕЙ РЕЛЕВАНТНОСТИ

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

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

1. Счетчик вхождений в прообразы;
2. Число вхождений в прообразы с малой мощностью;
3. Число вхождений в прообразы с высокой степенью истинности.

Однако, построение всего множества минимальных начальных прообразов имеет экспоненциальную вычислительную сложность и требует экспоненциального объема памяти для хранения прообразов. Решение этой проблемы заключается в том, чтобы не вычислять прообразы в явном виде (это возможно в силу того, 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 из рассмотрения — оставшиеся слои всегда имеют решение. Для построения решения необходимо обойти соответствующий путь, объединить *LP*-коды и упростить дерево истинности до числового значения.

Из того, что слой эквивалентен пути в дереве достижимости следует:

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

Оценка вычислительной сложности для задачи подсчета путей в графе — равна $O(S^3)$, где S — количество вершин в дереве достижимости.

При подсчете всевозможных путей необходимо 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 ветки и вернуться назад.

Таким образом, общая оценка сложности этой задачи равна $O(S^3M_0)$, где M_0 — количество начальных утверждений в базе знаний.

Верхняя оценка объема памяти стека *LP*-кодов будет равна $O(M_0S)$.

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 количества вхождений в прообразы с малой мощностью или высокой степенью истинности без построения самих прообразов, можно выполнить следующим способом:

2.1. 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 с помощью обхода дерева достижимости в глубину. При обходе дерева достижимости, с каждым правилом (ИЛИ-связью) необходимо связать 2 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.

2.2. 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 необходимо завершить. Если противоречий нет - необходимо обновить показатели релевантности для объектов из *LP*-кода текущей *ИЛИ*-вершины.

Вычислительная сложность выполняемого обхода зависит от количества 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* (то есть к числу вершин в графе базы знаний).

Если же степень реберной связности графа базы знаний близка к единице, то количество всевозможных путей с наилучшими оценками будет в худшем случае полиномиальным.

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

1. Если граф базы знаний имеет степень реберной связности близкую к числу вершин в данном графе, то сложность описанного метода будет экспоненциальной.
2. Если же граф базы знаний имеет степень реберной связности, близкую к единице, то оценка вычислительной сложности описываемого метода будет равна: $O(N) + O(M^2)+O(S^3M_0)+O(S^2P) = O(N+M^2+S^3M_0+S^2P)$, где

N - количество правил в базе знаний

M - количество вершин в графе базы знаний

M_0 - количество начальных утверждений в базе знаний

S - количество вершин в дереве достижимости

P - количество всевозможных путей в дереве достижимости, посещаемых при обходе ребер с наилучшими оценками.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Описанный метод, реализован в системе 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 по адресу: http://sourceforge.net/projects/lpexpert.

Система реализована на языке C++ с использованием библиотеки Qt4. Дерево достижимости реализовано в виде класса ReachabilityThree, который является контейнером для списка смежности. Узлы дерева истинности являются экземплярами класса TruthNode и размещаются в контейнерах заключений правил (элементы TRule::Perm).

Реализация описанных алгоритмов выполнена в классе LPGraph. При создании экземпляра данного класса, в конструктор передается ссылка на экземпляр контекста, содержащий все исходные данные.

Класс LPGraph предоставляет следующий интерфейс:

- Метод LPGraph::prepare() выполняет подготовку тестовой базы знаний для обработки.
- Метод LPGraph::build() выполняет построение дерева достижимости и дерева истинности.
- Метод LPGraph::getToAsk() выполняет вычисление показателей релевантности (способом, описанным в пункте 5) и возвращает индекс объекта, наиболее подходящего для конкретизации.

- Метод LPGraph::setValue(int object, const TValue & value) выполняет присваивание конкретного значения объекту базы знаний, после чего обновляет дерево достижимости и дерево истинности.
- Метод LPGraph::hasAnswer() возвращает индекс значения ответа, либо 0, если информации для вывода ответа недостаточно. Значение -1 соответствует ответу «решений нет».

ЗАКЛЮЧЕНИЕ

В статье рассмотрены теоретические особенности построения оптимизированного алгоритма LP-вывода, с поддержкой нечеткости на уровне коэффициентов доверия. Предложена основа для улучшения качества решений кластерно-релевантного LP-вывода.

Полезность практического применения разрабатываемого метода 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 [6].

Литература

1. Болотова С.Ю. Алгоритмы релевантного обратного вывода, основанные на решении продукционно-логических уравнений / С.Ю. Болотова, С.Д. Махортов // 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 / С. Д. Махортов // Программная инженерия. – 2010, № 2. – С. 15–21.
3. Levi G., Sirovich F. Generalized And/Or Graphs // Artificial Intelligence Journal. – Amsterdam : Elsevier, 1976. – № 7. – P. 243-259.

4. Махортов С. Д. Интегрированная среда логического программирования LPExpert / С.Д. Махортов // Информационные технологии. – М. : Новые технологии, 2009 г. – № 12. – С. 65–66.
5. Заде Л. А. Понятие лингвистической переменной и его применение к принятию приближенных решений : пер. с англ. / Л. А. Заде. – М. : Мир, 1976. – 165 с.
6. Чечкин А.В. Нейрокомпьютерная парадигма информатики / А.В. Чечкин // Нейрокомпьютеры: разработка, применение. 2011, №7. С. 3–9.