УДК 004.825

ИССЛЕДОВАНИЕ И РЕАЛИЗАЦИЯ АЛГОРИТМОВ ОБРАТНОГО ВЫВОДА В РАСПРЕДЕЛЕННЫХ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

М. В. Лещинская, С. Д. Махортов

Воронежский государственный университет

Поступила в редакцию 14.03.2019 г.

Аннотация. При создании, поддержке и эксплуатации больших баз знаний интеллектуальных систем не всегда в полной мере учитывается объем ресурсов, необходимый для их функционирования. В результате процессы работы с базами знаний, в частности, выяснение истинности гипотез, оказываются весьма ресурсозатратными. Усовершенствование алгоритмов логического вывода позволяет повысить эффективность работы интеллектуальных систем продукционного типа. Примером служит подход релевантного обратного вывода (LP-вывод), существенно уменьшающий число обращений к внешним источникам информации. Тем самым снижается ресурсоемкость логического вывода. Одна из стадий данного процесса состоит в вычислении специальных параметров, называемых показателями релевантности. В случае распределенных интеллектуальных систем эти стратегии должны учитывать распределенный характер баз знаний. Настоящая работа представляет основные понятия алгоритмов распределенного LP-вывода и рассматривает особенности стратегий подсчета показателей релевантности для распределенных LP-структур, применение которых значительно повышает эффективность работы распределенных интеллектуальных систем.

Ключевые слова: LP-структура, распределенная интеллектуальная система, релевантный обратный вывод, показатели релевантности.

Annotation. The amount of resources necessary for functioning of intelligent systems is not always fully taken into account while creating, maintaining and operating large knowledge bases of these systems. As a result, the processes of working with knowledge bases, in particular finding out the truth of hypotheses, turn out to be very resource-intensive. The improvement of logical inference algorithms allows to increase the efficiency of intellectual systems of production type. An example is the approach of the relevant backward inference (LP-inference), which significantly reduces the number of calls to external sources of information. This reduces the resource consumption of inference. One of the stages of this process is to calculate special parameters called relevance scores. In case of distributed intelligent systems these strategies should take into consideration the distributed nature of knowledge bases. This research presents the basic concepts of algorithms of distributed LP-inference and considers the peculiarities of strategies for calculating relevance scores for distributed LP-structures, the use of which significantly increases the efficiency of distributed intelligent systems.

Keywords: LP-structure, distributed intelligent system, relevant backward inference, relevance scores.

ВВЕДЕНИЕ

Интеллектуальные системы продукционного типа находят применение в различных предметных областях [1–2]. Однако для их работы на практике требуется значительный объем вычислительных ресурсов. В частности, логический вывод обычно сопровождается интенсивным обменом данными с внешней памятью или интерактивным пользователем.

В работе [3] был предложен новый метод обратного продукционно-логического выво-

да – *релевантный* LP-вывод. Он направлен на снижение числа обращений к внешним источникам информации. Для этой цели применяются алгебраические модели продукционных систем – LP-структуры [4]. В основе метода лежат нахождение решений продукционно-логического уравнения (как *прообразов* бинарного отношения) и исследование множества этих решений с помощью так называемых *показателей релевантности*. В результате истинный прообраз определяется за счет меньшего числа обращений к базе данных или интерактивному пользователю.

В работе [5] было введено понятие *распределенной* LP-структуры, открывающее возможности для распространения преимуществ разработанной ранее теории на распределенные интеллектуальные системы, в [6] – исследован класс уравнений в распределенной LP-структуре.

В силу распределенного характера интеллектуальной системы задача снижения числа запросов к внешнему источнику информации усложняется проблемой снижения трафика между узлами вычислительной сети. Таким образом, при разработке стратегии распределенного релевантного LP-вывода, наряду с известными ранее показателями релевантности [3], необходимо учитывать новые, связанные с особенностями распределенного характера системы.

В настоящей работе предлагаются и анализируются стратегии и алгоритмы подсчета релевантности в распределенном LP-выводе. Эвристически представленные алгоритмы позволяют уменьшить количество внешних запросов в процессе обратного вывода, а также снизить интенсивность обмена информацией между узлами вычислительной сети, хранящей базу знаний распределенной интеллектуальной системы.

1. МАТЕРИАЛЫ И МЕТОДЫ

1.1. LP-структуры и моделируемые ими продукционные системы

Под *LP-структурой* [4] подразумевается алгебраическая система, образованная математической решеткой F [7], на которой задано (дополнительное) бинарное отношение R, обладающее некоторыми («продукционно-логическими») свойствами.

Отношение R называется *продукционно-логическим*, если оно обладает рефлексивностью, то есть содержит все пары вида (a,a), транзитивностью и другими свойствами, которые определяются конкретной моделью. Одно из таких свойств – *дистрибутивность*. Неформально дистрибутивность отношения означает возможность логического вывода по частям и совмещения его результатов на основе решеточных операций.

Продукционные экспертные системы манипулируют множествами фактов и правил (продукций) [1]. Факт представляет некоторое суждение о внешнем мире. Продукционная система содержит *рабочую память*. Это некоторое подмножество фактов, которые на текущий момент считаются выполненными.

Правило (*продукция*) состоит из предпосылки и заключения. *Предпосылка* обычно представляет собой выражение над фактами. Она может быть выполненной (истинной) или невыполненной (ложной) при текущем состоянии рабочей памяти. *Заключение* – это действие, которое можно осуществить, если верна предпосылка (например, добавить к рабочей памяти новый факт). Совокупность правил называется *базой знаний*.

Обратный вывод по некоторому набору фактов – *гипотезе*, путем анализа правил в направлении от заключения к предпосылке, подтверждает или опровергает справедливость гипотезы при заданном исходном содержимом рабочей памяти. В процессе обратного вывода содержимое рабочей памяти также меняется.

Рассматриваемый подход к исследованию продукционных систем основан на представлении множеств фактов и правил LP-структурой. Каждый элементарный факт изображается атомом решетки, предпосылка и заключение правила – соответствующими элементами решетки, а правила представляются парами бинарного отношения R.

1.2. Распределенные LP-структуры

Перейдем к понятию распределенной LP-структуры [5]. Она моделирует распределенную интеллектуальную систему. Каждому атому решетки и каждой паре отношения R сопоставляется множество узлов «вычислительной сети», на которых они «хранятся». Пусть N – множество узлов, N – порожденный им булеан (множество всех подмножеств). Атомы решетки F и пары отношения R будут помечаться элементами решетки N как атрибутами.

Таким образом, на решетке определено отображение *Nodes*(), сопоставляющее каждому атому $a \in F$ единственный непустой элемент $X \in \mathrm{N}$ ($Nodes(a) = X$, $X \neq \varnothing$). Кроме того, каждой паре $(A, B) \in R$ также сопоставляется непустой элемент $Y \in \mathrm{N}$ ($Nodes(A, B) = Y$, $Y \neq \varnothing$). При заданном отображении *Nodes*() решетка F и отношение R считаются распределенными.

Описанная алгебраическая система называется *распределенной LP-структурой*.

Далее рассматриваются признаки «хорошего» распределения информации по узлам. В частности, считается целесообразным хранение на узле всех правил, в которых используются факты этого же узла. Для формализации такого требования в LP-структуре наряду с функцией *Nodes*() вводятся еще два отображения. Поскольку решетка F – атомно-порожденная [7], для любого ее элемента A существует представление $A = \bigcap_i a_i$ в виде объединения атомов. Обозначим $NodesMeet(A) = \bigcap_i Nodes(a_i)$; $NodesJoin(A) = \bigcup_i Nodes(a_i)$.

Отображение $NodesMeet()^i$ определяет узлы, каждый из которых содержит все атомы элемента A. Отображение *NodesJoin*() выдает все узлы, хранящие хотя бы один из таких атомов.

Распределенное бинарное отношение R на распределенной решетке F называется *релевантно-корректным*, если для любой $(A, B) \in R$ справедливо $Nodes(A, B) \supseteq NodesJoin(A) \bigcup NodesJoin(B)$; *релевантно-нормализованным*, если $Nodes(A, B) = NodesJoin(A) \bigcup NodesJoin(B)$.

1.3. О распределенном LP-выводе и показателях релевантности

В работе [6] введен аппарат продукционно-логических уравнений в распределенной LP-структуре, установлены их свойства и предложены методы решения. Процесс вычисления решения (множества начальных прообразов некоторой гипотезы) соответствует обратному логическому выводу. Для его завершения остается найти в множестве истинный прообраз, обращаясь за фактами к внешнему источнику.

Согласно [3], стратегия релевантного вывода направлена на минимизацию количества внешних запросов. Для этого эффективным является приоритетный просмотр прообразов, содержащих наиболее «релевантные» факты (атомы решетки). Таковыми в первую очередь считаются атомы, присутствующие в максимальном количестве построенных прообразов. Тогда единственный отрицательный ответ на заданный вопрос исключает из рассмотрения сразу большое количество прообразов. Второй показатель релевантности – присутствие атомов в прообразах минимальной мощности. Таким образом, предпочтение отдается тем прообразам, проверка истинности которых потребует меньшего количества вопросов пользователю (или обращений к базе данных).

Сочетая указанные два показателя релевантности, можно достичь результатов, существенно превышающих возможности стандартной машины вывода. Как показывают эксперименты [3], применяя LP-вывод, можно достичь снижения числа медленных запросов в среднем на 15–20 %.

2. РЕЗУЛЬТАТЫ И ИХ ОБСУЖДЕНИЯ

2.1. Новые параметры релевантности

Для *распределенной* интеллектуальной системы возникают дополнительные аспекты эффективного логического вывода. Появляется еще одна цель – снижение трафика между узлами вычислительной сети. Новые показатели релевантности связаны с атрибутами

хранения фактов и правил, то есть отображениями *Nodes*(), *NodesMeet*(), *NodesJoin*().

Как уже отмечалось, в [3] были введены следующие параметры релевантности исследуемых фактов (атомов), обеспечивающие оптимизацию обратного вывода: 1) присутствие атома в максимальном количестве прообразов гипотезы; 2) присутствие атома в прообразах минимальной мощности. В настоящей работе рассматриваются дополнительные параметры, а именно: 3) присутствие атома в прообразе A таком, что в $NodesMeet(A)$ содержится наибольшее количество узлов; 4) присутствие атома в прообразе A таком, что $NodesMeet(A) = NodesJoin(A)$.

Условие 3) гарантирует, что прообраз $A = \bigcap_i a_i$ имеет атомы a_i, которые в совокупности хранятся в наибольшем количестве узлов. Их приоритетный просмотр ускорит поиск истинного атома, и в то же время уменьшит трафик между узлами сети. Параметр 4) означает, что прообраз $A = \bigcap_i a_i$ лишь целиком хранится в некоторых узлах. Приоритетный просмотр таких атомов ускоряет поиск истинного прообраза, поскольку гарантированно снижается сетевой трафик.

Таким образом, для запуска LP-вывода требуется вычислить все начальные прообразы некоторой гипотезы G. Им соответствуют минимальные подмножества не выводимых из базы знаний фактов, которые порождают факты G. Далее среди прообразов достаточно найти хотя бы один истинный, то есть состоящей целиком из выполненных фактов (атомов), тогда гипотеза окажется верной.

С этой целью, прежде чем выяснять истинность какого-либо атома, найдем наиболее релевантный из них. Присвоив всем атомам нулевой приоритет, будем затем повышать его на 1, если атом содержится в максимальном (по сравнению с другими атомами) количестве прообразов. Далее учитываем присутствие атомов в прообразах минимальной мощности, повышая его приоритет на 1.

Примем также во внимание новые параметры релевантности. Например, повысим приоритет атома на 1, если он содержится в прообразе, отображение *NodesMeet*() которого содержит наибольшее количество узлов (он чаще встречается в узлах вычислительной сети).

Заметим, что атом, присутствующий в прообразе минимальной мощности, по-видимому, с большей степенью уверенности находится в таком элементе A, отображение $NodesMeet(A)$ которого содержит наибольшее количество узлов. Отсюда появляется шанс совместить учет параметров релевантности 2), 3).

2.2. Общий алгоритм

Разработана программная система, реализующая алгоритм распределенного LP-вывода, основанного на решении продукционно-логических уравнений вида $R^L(X) = b$ [6]. Атому $b \in F$ соответствует общее решение данного уравнения – множество $\{X\}$ всех минимальных начальных прообразов. На атомах решетки введем ряд частично определенных булевых функций.

1. *TrueData* (функция «истинности» для рабочей памяти), которую можно доопределять путем обращения к внешнему источнику информации ($TrueData(x) = 1$ – элемент x содержится в рабочей памяти, $TrueData(x) = 0$ – не содержится, $TrueData(x) = null$ – проверка x не производилась).

2. *TrueNode* (функция «истинности» для узлов вычислительной сети), которую можно доопределять путем обращения к узлам вычислительной сети ($TrueNode(x, u_i) = 1$ – элемент x хранится на узле u_i, $TrueNode(x, u_i) = 0$ – не хранится, $TrueNode(x, u_i) = null$ – проверка элемента x не производилась).

В работе [3] введены обозначения $dataT = \bigcup x_k (TrueData(x_k) = 1)$, $dataF = \bigcup x_k (TrueData(x_k) = 0)$. Дополним их новыми, учитывающими распределенный характер системы: $nodeT(u_i) = \bigcup x_k (TrueNode(x_k, u_i) = 1)$, $nodeF(u_i) = \bigcup x_k (TrueNode(x_k, u_i) = 0)$, где u_i – i-й узел вычислительной сети. Необходимо найти такой элемент $X^0 \in \{X\}$, что $X^0 \subseteq dataT \bigcup nodeT$ (если он существует). При решении этой задачи требуется минимизировать доопределения функций *TrueData* и *TrueNode*.

Листинг 1

```
X^0 = null
{X} = getPreImages(b)
while X^0 = null and {X} ≠ ∅ do
        k = getRelevantIndex({X}, dataT)                    (1)
        Ask(x_k)
        foreach X_j ∈ {X} do
            if X_j ⊆ dataT then
                X^0 = X_j
                break
            end
            if X_j ⋂ dataF ≠ ∅ then {X} = {X} \ X_j
        end
end
```

Листинг 2

```
int getRelevantIndex3({X}, dataT)
    // Инициализация приоритетов
    // начальных атомов решетки и
    // массива, соответствующего
    // количеству узлов
    // в отображении NodesMeet(A)
    foreach x_k ∈ F_0 do
        Priority[k] = 0
        Z[k] = kolNodesMeet(A_k)
    end
    // Вычисление приоритетов начальных атомов
    foreach x_k ∈ F_0 do
        if not x_k ⊆ dataT ⋃ dataF then
        // Рассматриваем лишь непроверенные атомы
            foreach X_j ∈ {X} do
                if x_k ⊆ X_j then
                    Priority[k] += Z[k]
        // Используя стратегию подсчета
        // релевантности, основанную на 3
        // параметре релевантности
                end
            end
        end
    end
    // Нахождение индекса наиболее релевантного атома
    int nRel = null
    foreach x_k ∈ F_0 do
        if nRel = null or Priority[k] > Priority[nRel] then nRel = k
    end
    return nRel
end
```

Листинг 3

Подсчет количества узлов, содержащихся в отображении $NodesMeet(A_i)$

```
int kolNodesMeet(A_i)
    // Для каждого узла U_j из множества узлов вычислительной сети U
    foreach U_j ∈ U do
            count = null // Общее количество атомов в прообразе
            countTrue = null // Количество атомов хранящихся на узле U_j
            // Для каждого атома из прообраза A_i
            foreach x_k ∈ A_i do
                    // Ищем атом x_k, хранящийся на узле U_j
                    if x_k ⊆ U_j then countTrue++
                    count++
            end
    // Если количество атомов из прообраза A_i,
    // хранящихся на узле U_j совпадает с общим
    // количеством атомов прообраза A_i, значит
    // такой узел будет содержаться в отображении
    // NodesMeet(A_i).
            if countTrue = count then kol++
    end
    return kol
end
```

Листинг 4

```
int getRelevantIndex4({X}, dataT)
    // Инициализация приоритетов начальных атомов решетки
    foreach x_k ∈ F_0 do
            Priority[k] = 0
            Meet[k] = kolNodesMeet(A_k)
            Join[k] = kolNodesJoin(A_k)
    end
    // Вычисление приоритетов начальных атомов
    foreach x_k ∈ F_0 do
            if not x_k ⊆ dataT ⋃ dataF then
                    // Рассматриваем лишь непроверенные атомы
                    foreach X_j ∈ {X} do
                            if x_k ⊆ X_j then
                                    if Meet[k] = Join[k] then Priority[k]++
    // Используя стратегию подсчета релевантности,
    // основанную на 4 параметре релевантности
                            end
                    end
            end
    end
    // Нахождение индекса наиболее релевантного атома
    int nRel = null
    foreach x_k ∈ F_0 do
            if nRel = null or Priority[k] > Priority[nRel] then nRel = k
    end
    return nRel
end
```

Листинг 5

Подсчет количества узлов, содержащихся в отображении $NodesJoin(A_i)$

```
int kolNodesJoin(A_i )
    // Для каждого узла U_j из множества узлов вычислительной сети U
    foreach U_j ∈ U do
        countTrue = null // Количество атомов хранящихся на узле U_j
        // Для каждого атома из прообраза A_i
        foreach x_k ∈ A_i do
            // Ищем атом x_k, хранящийся на узле U_j
            if x_k ⊆ U_j then
                countTrue++
            end
        end
    // Если количество атомов из прообраза A_i,
    // хранящихся на узле U_j больше нуля, то
    // значит такой узел будет содержаться в
    // отображении NodesJoin(A_i).
        if countTrue > 0 then kol++
    end
    return kol
end
```

В работе [3] представлен алгоритм релевантного LP-вывода (*Листинг 1*).

Здесь функция *getPreImages*(b) решает уравнение $R^L(X) = b$, то есть строит множество $\{X\}$ всех минимальных начальных прообразов атома b. Функция *getRelevantIndex*($\{X\}, dataT$), ищет индекс k любого из наиболее релевантных и ранее не проверенных на истинность атомов, содержащихся в элементах множества начальных прообразов $\{X\}$. $Ask(x_k)$ запрашивает внешний источник об истинности атома x и соответственно модифицирует множества $dataT$, $dataF$ (доопределяет функцию *TrueData*).

Для распределенного LP-вывода необходимо добавить в алгоритм функции, учитывающие распределенный характер LP-структуры: стратегии подсчета релевантности и обращения к узлам сети.

2.3. Алгоритмы стратегий подсчета релевантности

Стратегия релевантности ориентирована на общее снижение количества вызовов функции $Ask(x_k)$. С этой целью функция *getRelevantIndex* может оперировать упомянутыми в п. 5 показателями релевантности начальных атомов. Одна из возможных версий функции *getRelevantIndex* представлена ниже. Она использует 3-й параметр релевантности: присутствие атома в прообразе A таком, что в $NodesMeet(A)$ содержится наибольшее количество узлов (*Листинг 2*).

Функция $kolNodesMeet(A_k)$ отвечает за подсчет количества узлов, содержащихся в отображении $NodesMeet(A_k)$.

В *Листинге 3* приведен алгоритм, реализующий функцию $kolNodesMeet(A_i)$.

Если в *Листинге 1* вместо присваивания (1) указать $k = getRelevantIndex3(\{X\}, dataT)$, то получится LP-вывод, учитывающий при подсчете релевантности распределенный характер системы. При этом стратегия ориентирована на использование 3 параметра релевантности (п. 2.1).

Рассмотрим другую версию функции *getRelevantIndex*, опирающуюся на параметр релевантности 4, а именно – присутствие атома в прообразе A таком, что $NodesMeet(A) = NodesJoin(A)$ (Листинг 4).

Если в *Листинге 1* на месте строки (1) записать $k = getRelevantIndex4(\{X\}, dataT)$, получится LP-вывод, учитывающий при под-

счете релевантности распределенный характер системы. При этом стратегия подсчета релевантности опирается на 4 параметр релевантности (п. 2.1).

Эвристически ясно, что оба алгоритма (*getRelevantIndex*3, *getRelevantIndex*4) позволяют уменьшить количество внешних запросов в процессе обратного вывода, а также снизить трафик в узлах вычислительной сети.

ЗАКЛЮЧЕНИЕ

В настоящей работе проведено исследование возможностей практического применения теории распределенных LP-структур для повышения эффективности работы распределенных интеллектуальных систем продукционного типа. С этой точки зрения приведены уточнения функциональности распределенной LP-структуры.

С учетом особенностей систем рассматриваемого типа были предложены стратегии подсчета релевантности, которые позволят повысить быстродействие обратного вывода. Представлены также соответствующие алгоритмы.

В дальнейшем планируется рассмотреть особенности компьютерной реализации предложенных стратегий, а также экспериментально получить и исследовать статистические результаты алгоритмов распределенного LP-вывода, подтверждающие их эффективность.

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00037.

СПИСОК ЛИТЕРАТУРЫ

1. *Sowyer, B.* Programming Expert Systems in Pascal / B. Sowyer, D. Foster. – John Wiley & Sons, Inc., 1986. – 198 p.

2. *Maciol, A.* An application of rule-based tool in attributive logic for business rules modeling / A. Maciol // Expert Systems with Applications. – April 2008. – Vol. 34, №. 3. – P. 1825 –1836.

3. *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. – No 2. – С. 40–50.

4. *Махортов, С. Д.* Математические основы искусственного интеллекта: теория LP- структур для построения и исследования моделей знаний продукционного типа / С. Д. Махортов ; под ред. В. А. Васенина. – М. : Изд-во МЦНМО, 2009. – 299 с.

5. *Махортов, С. Д.* Алгебраическая модель распределенной логической системы продукционного типа / С. Д. Махортов // Программная инженерия. – 2015. – No 12. – С. 32–38.

6. *Махортов, С. Д.* Продукционно-логические уравнения в распределенной LP-структуре / С. Д. Махортов // Программная инженерия. – 2016. – No 7. – С. 324–329.

7. *Биркгоф, Г.* Теория решеток : пер. с англ. / Г. Биркгоф. – М. : Наука, 1984. – 568 с

Лещинская Мария Владимировна – студентка 2 курса магистратуры факультета прикладной математики, информатики и механики Воронежского государственного университета.
Email: maria-leshchinskaya@mail.ru

Махортов Сергей Дмитриевич – д-р физ.-мат. наук, заведующий кафедрой программирования и информационных технологий факультета компьютерных наук Воронежского государственного университета.
Email: msd_exp@outlook.com.

Leshchinskaya Maria Vladimirovna – 2nd year master's student at Applied Mathematics, Informatics and Mechanics department, Voronezh State University.
Email: maria-leshchinskaya@mail.ru

Makhortov Sergey Dmitrievich – doctor of physical and mathematical sciences, head of the department of programming and information technology, faculty of Computer Science, VSU.
Email: msd_exp@outlook.com