УДК 519.688

РЕАЛИЗАЦИЯ И ИССЛЕДОВАНИЕ ПАРАЛЛЕЛЬНОГО РЕЛЕВАНТНОГО ВЫВОДА

С. Ю. Болотова

Воронежский государственный университет

Поступила в редакцию 27.09.2017 г.

Аннотация. Релевантный LP-вывод, основанный на решении логических уравнений, является эффективным для исследования и верификации продукционно-логических систем. Он позволяет существенно уменьшить число выполняемых запросов к внешнему источнику информации. Однако, при больших объемах баз знаний и их сложной структуре для осуществления вывода может потребоваться значительное количество времени. В работе описывается реализация алгоритма усовершенствованного LP-вывода, основанного на применении параллельных вычислений, приводятся соответствующие псевдокоды и графы вычислений. Эффективность предлагаемого подхода демонстрируется с помощью множественных экспериментов, результаты которых обрабатываются с привлечением методов математической статистики.
Ключевые слова: релевантный вывод, логические уравнения, параллельные алгоритмы, статистические исследования.
Annotation. The relevant LP-inference, which is based on the solution of logical equations, is an effective tool that can be used for research and optimization of production-logical systems. It allows to significantly reduce the number of external queries. However, the processing of large knowledge bases and their "deep" structure may require an excessive amount of computational resources of the computer. The paper describes the advanced LP-inference algorithm implementation based on the parallel computations, as well as the corresponding pseudo-codes and computing graphs. The benefits of parallel LP-inference are confirmed experimentally using mathematical statistics methods.
Keywords: backward inference, production systems, LP-structure, parallel computing.

ВВЕДЕНИЕ

Продукционно-логические системы являются важным направлением теоретических и прикладных исследований в области искусственного интеллекта [1]. Однако, несмотря на высокую практическую востребованность, продукционные системы весьма ресурсоемки. В частности, они зачастую требуют экспоненциального объема вычислений и интенсивного обмена с внешней памятью. Этот факт приводит к снижению их эффективности. Существуют предметные области, где необходимы высокая производительность и отклик в режиме реального времени. Данное обстоятельство препятствует применению продукционных систем. Снижение производительности оказывает существенное влияние и на уровень проводимых теоретических исследований. Следовательно, существенное повышение производительности продукционных и подобных им систем имеет важное значение для теоретических и прикладных исследований.

В монографии [2] была рассмотрена модель обратного вывода, основанная на решении продукционно-логических уравнений в специальных алгебраических системах – LP-структурах. Настоящая работа посвящена вопросам распараллеливания алгоритмов LP-вывода. Предлагаемая реализация позволяет уменьшить время нахождения решения в среднем на 25 %. Однако формальное обоснование эффективности предложенных алгоритмов представляет математически недоопределенную задачу, существенно зависящую от исходных данных. Поэтому преимущества алгоритмов LP-вывода подтверждают-

ся экспериментально. С учетом данного обстоятельства приобретает особое значение формальное сравнительное исследование результатов проводимых экспериментов. Аппаратом такого исследования может служить математическая статистика.

В работе приводятся результаты статистической обработки данных, полученных выполнением множества тестов параллельного релевантного обратного вывода.

ОБЩЕЕ ОПИСАНИЕ АЛГОРИТМА

Рассматриваемая программная система реализует алгоритм параллельного релевантного обратного вывода, основанного на решении продукционно-логических уравнений. Для этой цели применяются математически обоснованные в [2] механизмы нахождения истинного прообраза и ускорения обратного вывода. Здесь используются обозначения и определения, принятые в указанной работе.

LP-структурой (Lattice Production Structure) называется решетка с дополнительно заданным на ней бинарным отношением, которое обладает набором продукционно-логических свойств [2].

Стратегия релевантного вывода, основанная на решении уравнений в LP-структурах, направлена на минимизацию количества медленно выполняемых запросов (к базе данных или интерактивному пользователю). Обратный вывод начинается с построения всех минимальных начальных прообразов в LP-структуре для атомов, соответствующих значениям объекта экспертизы. Далее в построенном множестве достаточно найти тот прообраз, который содержит лишь истинные факты, после чего сразу можно сделать заключение о соответствующем значении объекта экспертизы. Эффективным в этом плане является приоритетный просмотр прообразов, содержащих значения наиболее «релевантных» объектов. Отрицательный ответ на запрос исключает последующие запросы об элементах связанного подмножества прообразов. Кроме того, при LP-выводе предпочтение отдается тестированию множеств фактов минимальной мощности.

Однако эксперименты показали, что при больших объемах баз знаний и их достаточно «глубокой» структуре процесс построения всех минимальных начальных прообразов может потребовать чрезмерного объема времени. В связи с данным обстоятельством метод релевантного LP-вывода был модифицирован. Параллельный релевантный LP-вывод предполагает одновременное построение набора начальных прообразов с их дальнейшим исследованием в разных потоках. Здесь и далее под «потоками» подразумеваются threads –потоки выполнения [4].

Рассмотрим вопрос нахождения истинного прообраза. Пусть даны конечная атомно-порожденная решетка F (атомами изображаются элементарные факты) и на ней бинарное отношение R (представляет совокупность продукционных правил). Отношение R – каноническое, то есть задано множеством пар вида (A, a), где $A \in F$, a – атом F.

Пусть выбран атом $b \in F$. Ему соответствует общее решение продукционно-логического уравнения с гипотезой в правой части – множество $\{X\}$ всех его минимальных начальных прообразов в логическом замыкании отношения R. Пусть также на множестве атомов решетки частично определена булева функция $True$ (функция «истинности»), которая способна «доопределяться» путем обращения к внешнему источнику информации. В моделируемой продукционной системе интерпретация данной функции такова:

- $True(X) = 1$, если соответствующий факт x содержится в рабочей памяти;
- $True(X) = 0$, если достоверно известно, что x не может содержаться в рабочей памяти;
- $True(X) = null$, если проверка x еще не производилась.

Введем обозначения $T = \bigcup x_k (True(x_k) = 1)$; $F = \bigcup x_j (True(x_j) = 0)$. Необходимо найти такой элемент $X^0 \in \{X\}$, что $X^0 \subseteq T$ (если он существует). В процессе решения задачи требуется также обойтись как можно меньшим количеством «доопределений» функции $True$.

Приведем алгоритм релевантного LP-вывода (*Алгоритм 1*).

Алгоритм 1

// Релевантный LP-вывод
$X^0 = null$
$\{X\} = getPreImages(b)$
while $X^0 = null$ **and** $\{X\} \neq \varnothing$ **do**
  $k = getRelevantIndex(\{X\}, T)$
  $Ask(x_k)$
  foreach $X_j \in \{X\}$ **do**
    if $X_j \subseteq T$ **then** $X^0 = X_j$
  break
  end
  if $X_j \bigcap F \neq \varnothing$ **then** $\{X\} = \{X\} \setminus X_j$
end
end

Используемая здесь функция $Ask(x)$ спрашивает внешний источник об истинности атома x и в соответствии с ответом модифицирует множества T и F (то есть «доопределяет» функцию $True$). Функция $getPreImages(b)$ решает продукционно-логическое уравнение с гипотезой b в правой части, то есть строит множество $\{X\}$ всех *минимальных* начальных прообразов для атома b.

Для решения уравнения исходное каноническое отношение R представляется в виде слоев [2], каждый из которых порождает не более одного решения. Слой содержит максимально возможный набор пар исходного отношения с уникальными правыми частями, два слоя различаются хотя бы одной парой. Непосредственная реализация слоев привела бы к избыточному хранению пар, так как слои могут иметь большие пересечения. Для решения вопроса представления слоев поступим следующим образом.

Разобьем R на непересекающиеся подмножества, каждое из которых образовано парами вида (A, x_p) с одним и тем же точечным элементом x_p в качестве правой части. Обозначим эти подмножества R^p соответственно их элементу x_p, $p \in P$. При реализации канонического отношения R для каждого x_p достаточно хранить лишь совокупность левых частей пар с правой частью x_p. Каждому из подмножеств R^p сопоставим итератор – индексную переменную j_p, способную перебирать собственное подмножество, останавливаясь на каждой паре ровно один раз. Таким образом, любой слой R_t в отношении R получается соответствующим набором значений итераторов $\{j_p\}$. Нахождение решения в отдельном слое сводится к получению списка начальных вершин графа, из которых достижима данная вершина (она соответствует атому правой части уравнения). Работа с различными слоями организована независимо и параллельно.

Первичный поток приложения создает новые потоки (они ограничены параметром *MaxThread* – максимальное количество потоков), передавая им пакет данных. Созданный поток находит решение продукционно-логического уравнения в отдельном слое. Получая начальную порцию прообразов, модуль LP-вывода сразу исследует их на предмет «истинности», обращаясь при необходимости за фактами к внешнему источнику. Если при обработке очередного прообраза он не оказывается «истинным», то вычисляется следующий прообраз. При этом запоминается информация об установленных на предыдущем шаге ложных фактах. На ее основе перед очередным процессом решения уравнения сужается множество актуальных правил. Это обстоятельство также существенно ускоряет работу. По окончании работы потоки завершаются.

В случае, когда рабочих потоков бывает слишком много, эффективность параллельных вычислений начинает снижаться. Частое создание и завершение потоков с малым временем работы, а также большое количество переключений их контекстов, увеличивают объем ресурсов. Поэтому при реализации параллельного LP-вывода используется заранее создаваемый пул потоков [4]. Максимальный размер пула ограничивается параметром. Его значение должно быть большим, чем число процессоров, чтобы добиться максимальной степени одновременности.

Главный поток выбирает поток из пула и передает ему необходимые данные для обработки. Когда количество активных потоков достигает максимума, запрос помещается в очередь. Если число активных потоков меньше максимального, создается новый поток, который получает пакет данных на обработку. Если же количество активных потоков

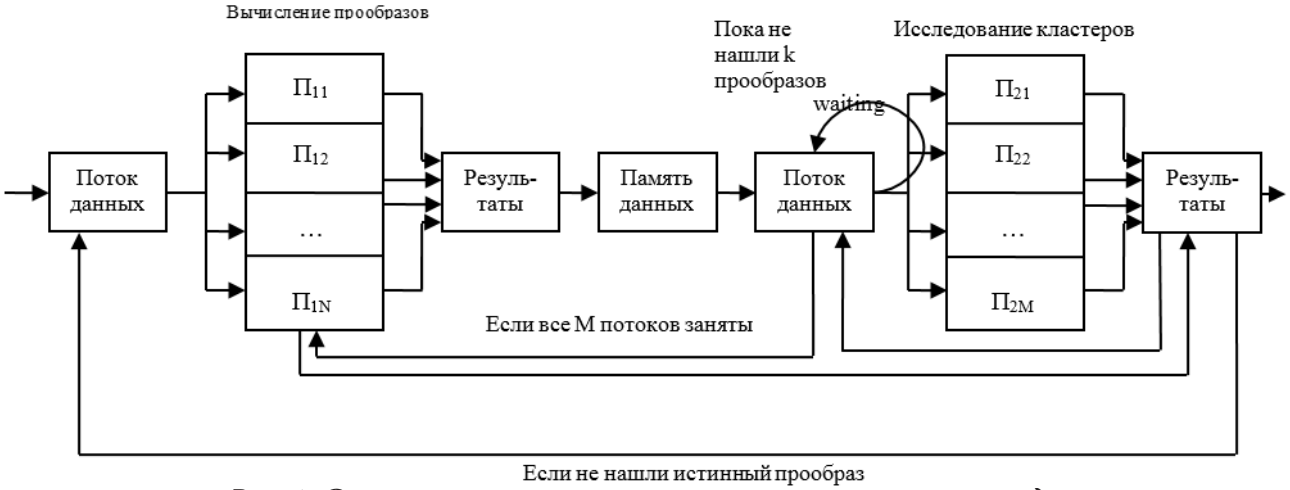

Рис. 1. Схема алгоритма параллельного релевантного вывода

равно максимальному, пакет ставится в очередь и ждет освобождения одного из потоков. В качестве механизма синхронизации потоков могут быть использованы критические секции, которые инициируются в процессе активизации.

С целью повышения производительности процессы вычисления прообразов и их дальнейшего исследования выполняются асинхронно. Пока решение не найдено, полученный на очередном шаге кластер прообразов помещается в динамическую структуру данных *Q*, организованную в виде очереди.

Главный поток извлекает кластеры прообразов из очереди и, при наличии свободных вторичных потоков выполнения, инициирует параллельное исследование их элементов на релевантность.

Найденные на очередном этапе наиболее релевантные элементы помещаются в очередь с приоритетом P, реализованную на основе двоичной кучи. Максимальный размер такой очереди задается параметром *maxRelevanceQueueSize*. Подобная организация данных позволяет отсортировать исследованные элементы в соответствии с их релевантностью, при необходимости изменять их порядок следования при добавлении новых элементов и за константное время получать доступ к элементу с максимальным показателем. Процесс определения релевантных объектов построенных кластеров продолжается до тех пор, пока решение не будет найдено.

Общая структура алгоритма параллельного релевантного вывода может быть представлена в виде следующей схемы (рис. 1).

Ниже приведен алгоритм параллельного релевантного вывода (*Алгоритм 2*).

В функции *getPreImagesCluster (b)* инициируется процесс вычисления фиксированного набора минимальных начальных прообразов атома b. Как уже было сказано, этот процесс осуществляется в отдельных потоках для обеспечения наибольшей эффективности.

Функция *getRelevantIndex* $(\{Y\}, T, relevance = 0)$ в отдельном потоке для каждого кластера находит релевантный элемент из нерассмотренных на данный момент. Она возвращает его показатель релевантности в переменной *relevance*. Далее элемент с показателем релевантности помещается в очередь с приоритетом. Если записываемый в очередь элемент уже находится в ней, то показатели релевантности складываются, и очередь перестраивается.

Параллельно с описанными действиями функция *Ask* (y_k) запрашивает внешний источник об истинности наиболее релевантного факта, извлеченного из очереди с приоритетом. При получении отрицательного ответа на запрос прообразы, содержащие элемент y_k, исключаются из рассмотрения, а релевантности уже исследованных элементов модифицируются.

При создании алгоритмов решения сложных задач с использованием параллельных вычислений очень важно верно оценить эффективность их использования, то есть полу-

Алгоритм 2

```
// Параллельный релевантный вывод
X^0 = null
while X^0 = null do
  asynchronous call {X} = getPreImagesCluster(b)
  if {X} ≠ ∅ then
    addClusterToQueue(Q, {X}) // Добавляем кластер в очередь Q
  end
  while not clustersQueueIsEmpty(Q) do in parallel // Если очередь не пуста
    {Y} = extractClusterFromQueue(Q) // Извлекаем кластер из очереди
    foreach (Y_j ∈ {Y}) do
      if Y_j ⋂ F ≠ ∅ then {Y} = {Y} \ Y_j
    end
    while X^0 = null do
      asynchronous if (relevanceQueueSize(P) < maxRelevanceQueueSize)
      then in parallel // Если в P есть свободное место
        // Индекс k релевантного объекта и его показатель relevance
        foreach (Y_j ∈ {Y}) do
          if Y_j ⋂ F ≠ ∅ then {Y} = {Y} \ Y_j
        end
        k = getRelevantIndex ({Y}, T, relevance = 0)
        InsertToQueue(P, k, relevance) // Объект в очередь с приоритетом
      else
        Waiting() // Ждем, пока не освободится место в очереди
      end
      asynchronous while (relevanceQueueSize(P)! = 0) do
        k = ExtractMaxFromQueue(P) // Эл-т с макс. релевантностью
        Ask (y_k) // Определяем истинность k-го факта
        foreach Y_j ∈ {Y} do
          if Y_j ⊆ T then
            X^0 = Y_j
            break
          end
          if Y_j ⋂ F ≠ ∅ then
            foreach (y_m ∈ {Y_j}) do
              // Эл-т в очереди P – понижаем релевантность
              if elemIsMemberOfQueue(y_m, P) then changeRelevance(y_m, P)
            end
          end
        end
      end
    end
  end
end
```

чаемое ускорение работы. С этой целью можно использовать модель вычислений в виде ациклического графа $G = (V, R)$, в котором множество операций, выполняемых в исследуемом алгоритме решения задачи, представляются, как множество вершин $V = \{1, \ldots, |V|\}$, а информационные зависимости между операциями – в виде множества дуг R [5]. Причем

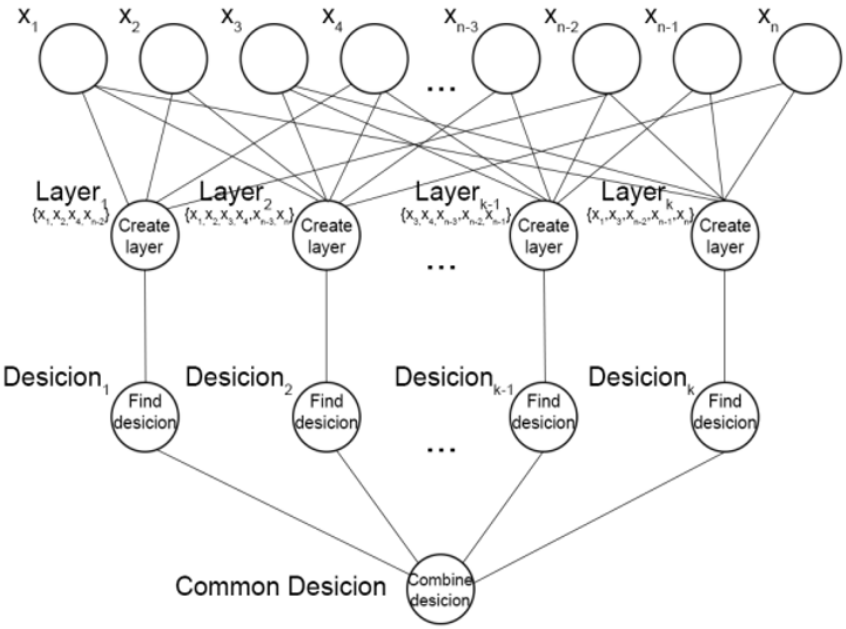

Рис. 2. Граф вычислений

дуга $r = (i, j)$ означает, что операция j использует результат выполнения другой операции i, а те операции алгоритма, между которыми нет пути, могут быть распараллелены.

Возможный способ описания параллельного выполнения алгоритма нахождения решения в отдельном слое с помощью графа изображен на рис. 2.

Ниже приведен алгоритм соответствующей функции $getPreImages\,Cluster(b)$ (*Алгоритм 3*). Обозначим множество слоев $\{R_i\}$ через R'. Как уже говорилось, решение в каждом слое вычисляется отдельным потоком. Количество активных в данный момент потоков хранится в переменной *countUsedThreads*. В случае, если в пуле есть свободный поток, он запускается и в нем вызывается функция $FindEquationSolution(R_i)$, которая находит решение уравнения в очередном слое R_i.

Алгоритм 3

```
// Нахождение решения уравнения на каждом
// слое в отдельном потоке
foreach R_i ∈ R' do in parallel
  if countUsedThreads < MaxThreads then
    BeginThread() // начать выполнение потока
    countUsedThreads++
    FindEquationSolution(R_i)
    ExitThread() // завершить поток
    countUsedThreads--
  else
    Waiting() // ждем освобождения потока
  end
end.
```

Функция *getRelevantIndex* ($\{Y\}$, T, *relevance*) находит индекс k любого из наиболее релевантных и ранее не проверенных на истинность атомов, содержащихся в элементах текущего кластера прообразов $\{Y\}$ и его показатель релевантности *relevance*. При имеющемся обычно большом количестве прообразов процесс выявления релевантных объектов весьма ресурсозатратен, поэтому он также распараллеливается. Для очередного кластера прообразов в отдельном потоке (количество потоков аналогично ограничено параметром *MaxThreadsCount*) запускается процесс их релевантного исследования на предмет истинности. Количество активных в данный момент потоков хранится в переменной *usedThreadsNumber*.. Для синхронизации потоков используется механизм критических секций. Приведем граф вычислений для алгоритма вычисления показателя релевантности (рис. 3).

Ниже представлен один из возможных параллельных вариантов функции *getRelevantIndex* (*Алгоритм 4*)

Используемая в этом алгоритме функция $MostRelevantFind(relevance)$ позволяет найти индекс k наиболее релевантного объекта

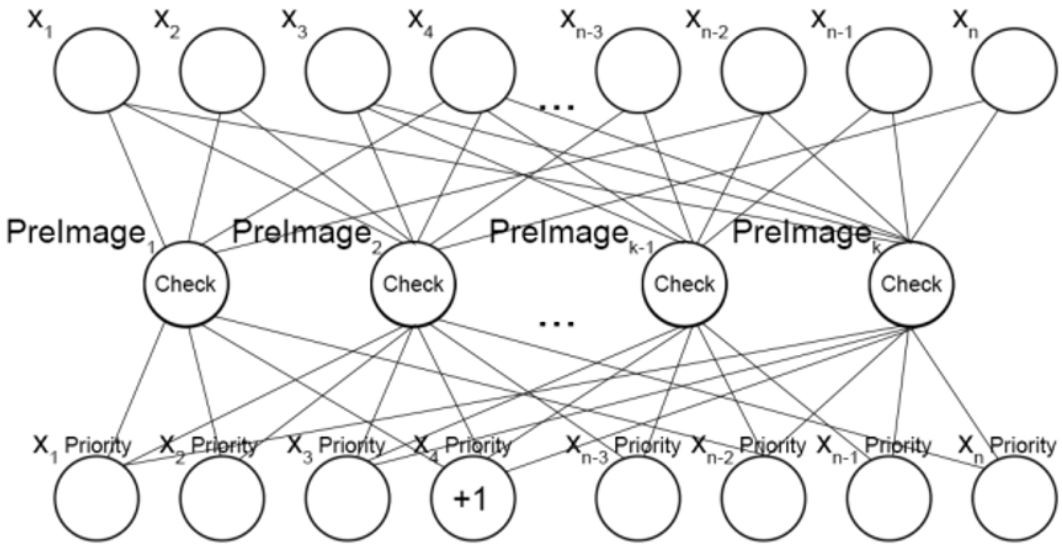

Рис. 3. Граф вычислений

Алгоритм 4

```
// Параллельный подсчет релевантности
int getRelevantIndex ({Y}, T, relevance)
if (usedThreadsNumber < MaxThreadsCount ) or  (countUsedThreads < MaxThreads ) then
  BeginThread() // начать выполнение потока
  if (usedThreadsNumber < MaxThreadsCount ) then
    usedThreadsNumber++
    usedSelfThreads = true
  else
    countUsedThreads++
    usedSelfThreads = false
  end
   k = MostRelevantFind(relevance) // Индекс релевантного атома
  ExitThread() // завершить поток
  if usedSelfThreads then
    usedThreadsNumber--
  else
    countUsedThreads--
  end
 return k
else
  Waiting() // Ждем, пока поток не освободится
end
```

и запомнить его показатель релевантности в переменной *relevance*.

РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Для объективной демонстрации возможностей параллельного релевантного LP-вывода было создано порядка пятисот баз знаний, существенно отличающихся по объему множеств правил. При тестировании получены показатели времени выполнения алгоритмов обычного и параллельного релевантного обратного вывода. Максимальное количество потоков также ограничивалось в диапазоне от 1 до 20 с шагом 3. Результаты проведенных экспертиз обработаны в пакете Statistica 6.

Для сравнения методов релевантного и параллельного релевантного обратного вывода

по результатам экспериментов были найдены дисперсии генеральных совокупностей. Более эффективным считается метод, который обеспечивает наименьшую дисперсию [7].

Проверим предположение о равенстве дисперсий δ_X^2 и δ_Y^2 с помощью F-критерия Фишера. Порядок применения F-критерия описан в [7].

Для сравнения рассматриваемых алгоритмов возьмем две выборки, объемы которых $n_1 = n_2 = 50$. Выборки представляют собой данные о времени выполнения алгоритмов обычного релевантного вывода и параллельного вывода (показано на примере с 5 потоками). Вначале определим, обладают ли эти методы одинаковой эффективностью $H_0 : \delta_X^2 = \delta_Y^2$ при уровне значимости $\alpha = 0.01$, если в качестве конкурирующей гипотезы принять предположение о том, что эффективность метода параллельного вывода выше, то есть $H_1 : \delta_X^2 > \delta_Y^2$.

Найдем выборочные дисперсии и исправленные выборочные дисперсии генеральных совокупностей данных.

$$S_u^2 = \frac{\sum u_i^2 - \left|\sum u_i\right|^2 / n_1}{n_1 - 1} = 2.47 \quad \text{и}$$

$$S_v^2 = \frac{\sum v_i^2 - \left|\sum v_i\right|^2 / n_2}{n_2 - 1} = 1.59.$$

Сравним дисперсии, вычислив их отношение по формуле $F_{набл} = \frac{S_X^2}{S_Y^2} = \frac{S_u^2}{S_v^2} = 1.553.$

По условию конкурирующая гипотеза имеет вид $\delta_X^2 > \delta_Y^2$, поэтому критическая область односторонняя, и при отыскании критической точки следует брать уровни значимости равными $\alpha = 0.01$. При числе степеней свободы $v_1 = n_1 - 1 = 49$, $v_2 = n_2 - 1 = 49$ находим критическую точку $F_{кр} = 1.295$.

Так как $F_{набл} > F_{кр}$, то гипотезу о равенстве дисперсий отвергаем – дисперсии различаются значимо. Следовательно, имеет место альтернативная гипотеза о том, что параллельный релевантный метод обратного вывода обеспечивают большую эффективность по времени выполнения.

Также был применен метод сравнения средних двух нормальных генеральных совокупностей. Для решения этой задачи в случае распределений, близких к нормальному, используется t-тест Стьюдента [7]. При том же уровне значимости 0,01 установим, значимо или незначимо различаются результаты исследований, в предположении, что они распределены нормально.

Проверяется гипотеза $H_0 : a_1 = a_2$ при альтернативной гипотезе $H_1 : a_1 > a_2$. Вычислим оценки средних и дисперсий: $\overline{x}_1 = 2.28$; $\overline{x}_2 = 1.85$; $s_1^2 = 2.47$; $s_2^2 = 1.59$.

Предварительно проверим гипотезу о равенстве дисперсий $H_0 : \delta_1^2 = \delta_2^2 : \frac{s_1^2}{s_2^2} = 1.553$. Поскольку $F_{кр} = 1.295$, то гипотеза о равенстве дисперсий отклоняется. Вычислим выборочное значение статистики критерия: $t = 1.49$. Число степеней свободы $k = 98$.

Так как по таблице критических точек распределения Стьюдента $t_{кр} = 0.992$, гипотеза о равенстве средних отклоняется и принимается альтернативная гипотеза. Другими словами, средние результаты измерений различаются значимо.

В представленных выше методах необходимо, чтобы случайные величины были распределены нормально. Однако специальные исследования [7] показали, что предложенные алгоритмы весьма устойчивы (особенно при больших объемах выборок) по отношению к отклонению от нормального распределения.

В процессе исследования данных в пакете Statistica были проведены однофакторный и двухфакторный дисперсионные анализы.

Levene's Test for Homogeneity of Variances (Spreadsheet7.sta)
Effect: Потоки
Degrees of freedom for all F's: 5, 294

	MS Effect	MS Error	F	p
Время	3,228410	0,609062	5,300625	0,000111

Рис. 4. Применение теста Левена

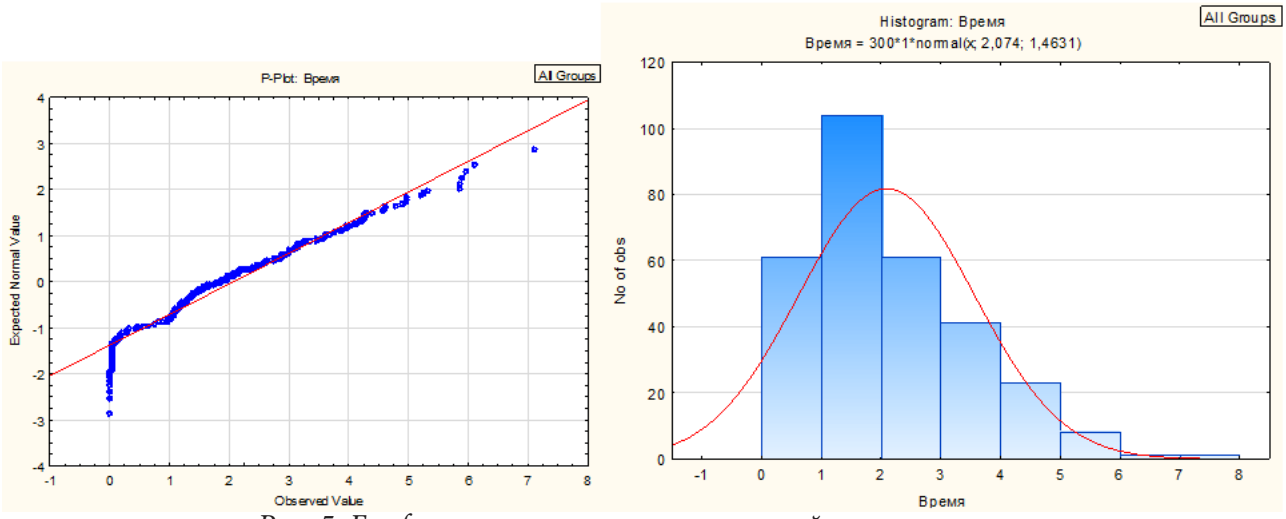

Рис. 5. График нормальных вероятностей и гистограмма

Univariate Tests of Significance for Время (Spreadsheet7.sta)
Sigma-restricted parameterization
Effective hypothesis decomposition

Effect	SS	Degr. of Freedom	MS	F	p
Intercept	1290,503	1	1290,503	632,6348	0,000000
Потоки	40,327	5	8,065	3,9538	0,001742
Error	599,727	294	2,040		

Рис. 6. Результат дисперсионного анализа

Tukey HSD test; variable Время (Spreadsheet7.sta)
Approximate Probabilities for Post Hoc Tests
Error: Between MS = 2,0399, df = 294,00

Cell No.	Потоки	{1} 2,2824	{2} 2,0739	{3} 1,8466	{4} 1,4901	{5} 2,0687	{6} 2,6826
1	1		0,978298	0,647656	0,061704	0,975810	0,726543
2	2	0,978298		0,968346	0,317385	1,000000	0,271218
3	5	0,647656	0,968346		0,812921	0,971366	0,040124
4	8	0,061704	0,317385	0,812921		0,327529	0,000440
5	10	0,975810	1,000000	0,971366	0,327529		0,262113
6	15	0,726543	0,271218	0,040124	0,000440	0,262113	

Рис. 7. Результаты анализа с помощью метода Тьюки

Проверка однородности групповых дисперсий выполнена с помощью теста Левена (рис. 4). Результаты показывают, что дисперсии не различаются ($p \ll 0{,}05$).

Следовательно, одно из условий применения параметрического варианта дисперсионного анализа не выполнено. Однако этот метод весьма устойчив в случае отклонения от указанного требования, поэтому можно продолжить исследование.

Для проверки нормальности распределения анализируемых данных были построены графики нормальных вероятностей и гистограммы, приведенные на рис. 5, из которых видно, что распределение близко к нормальному. Кроме того, объемы выборок позволяют продолжить исследование. Результат дисперсионного анализа представлен на рис. 6. Поскольку величина ошибки для нулевой гипотезы об отсутствии связи между числом пото- ков и временем выполнения $p \ll 0{,}05$, можно заключить, что время статистически значимо различается в зависимости от числа потоков. Апостериорный анализ с помощью метода Тьюки (рис. 7) показал, что статистически значимая разница во времени выполнения существует между парами параллельного релевантного вывода с 5 потоками и 15 пото-

Univariate Tests of Significance for Время (Spreadsheet7.sta)
Sigma-restricted parameterization
Effective hypothesis decomposition

Effect	SS	Degr. of Freedom	MS	F	p
Intercept	1290,503	1	1290,503	12592,53	0,00
Факты	550,367	9	61,152	596,71	0,00
Потоки	40,327	5	8,065	78,70	0,00
Факты*Потоки	24,764	45	0,550	5,37	0,00
Error	24,596	240	0,102		

Рис. 8. Результаты двухфакторного дисперсионного анализа

Kruskal-Wallis ANOVA by Ranks; Время (Spreadsheet7.sta)
Independent (grouping) variable: Потоки
Kruskal-Wallis test: H (5, N= 300) =14,42252 p =,0131

Depend.: Время	Code	Valid N	Sum of Ranks	Mean Rank
1	1	50	8124,500	162,4900
2	2	50	7609,000	152,1800
5	5	50	6967,000	139,3400
8	8	50	5860,500	117,2100
10	10	50	7664,500	153,2900
15	15	50	8924,500	178,4900

Рис. 9. Результаты анализы Краскела-Уоллиса

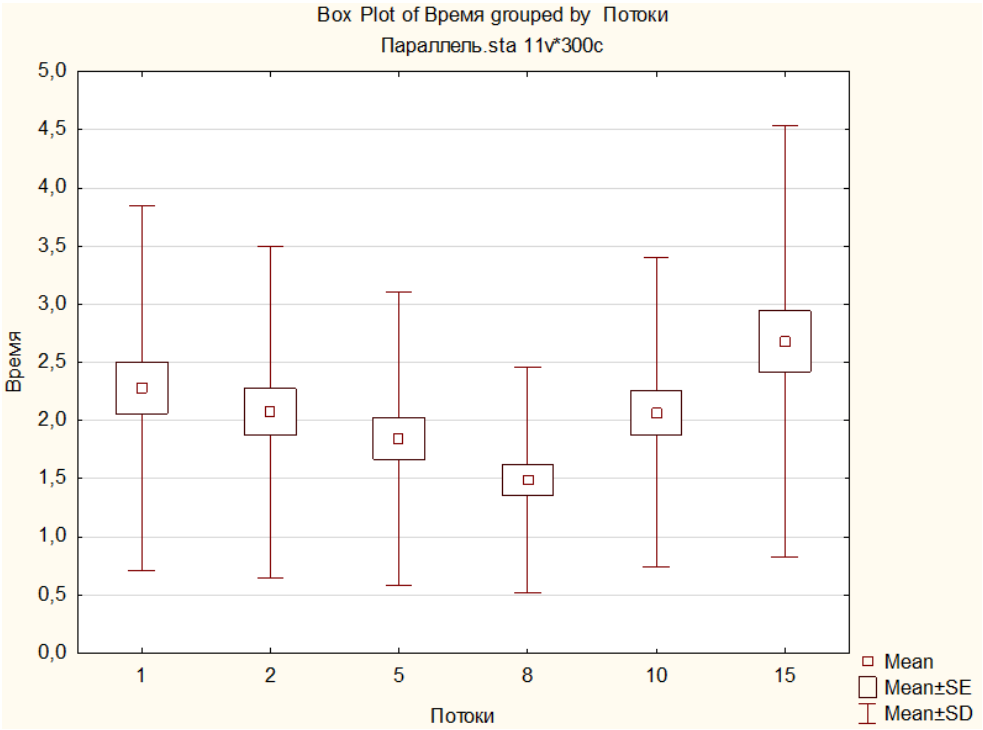

Рис. 10. Диаграммы размаха

ками, а также параллельного релевантного вывода с 8 и 15 потоками.

Двухфакторный дисперсионный анализ (рис. 8) выявил значительное влияние числа потоков и количества фактов, а также их взаимное воздействие, на время выполнения ($p \ll 0{,}05$).

Непараметрический однофакторный дисперсионный анализ Краскела-Уоллиса (рис. 9) продемонстрировал показал наличие статистически значимых различий между сравниваемыми группами ($p \ll 0{,}05$).

Для наглядности построены также диаграмма размаха (рис. 10) и графики зависимо-

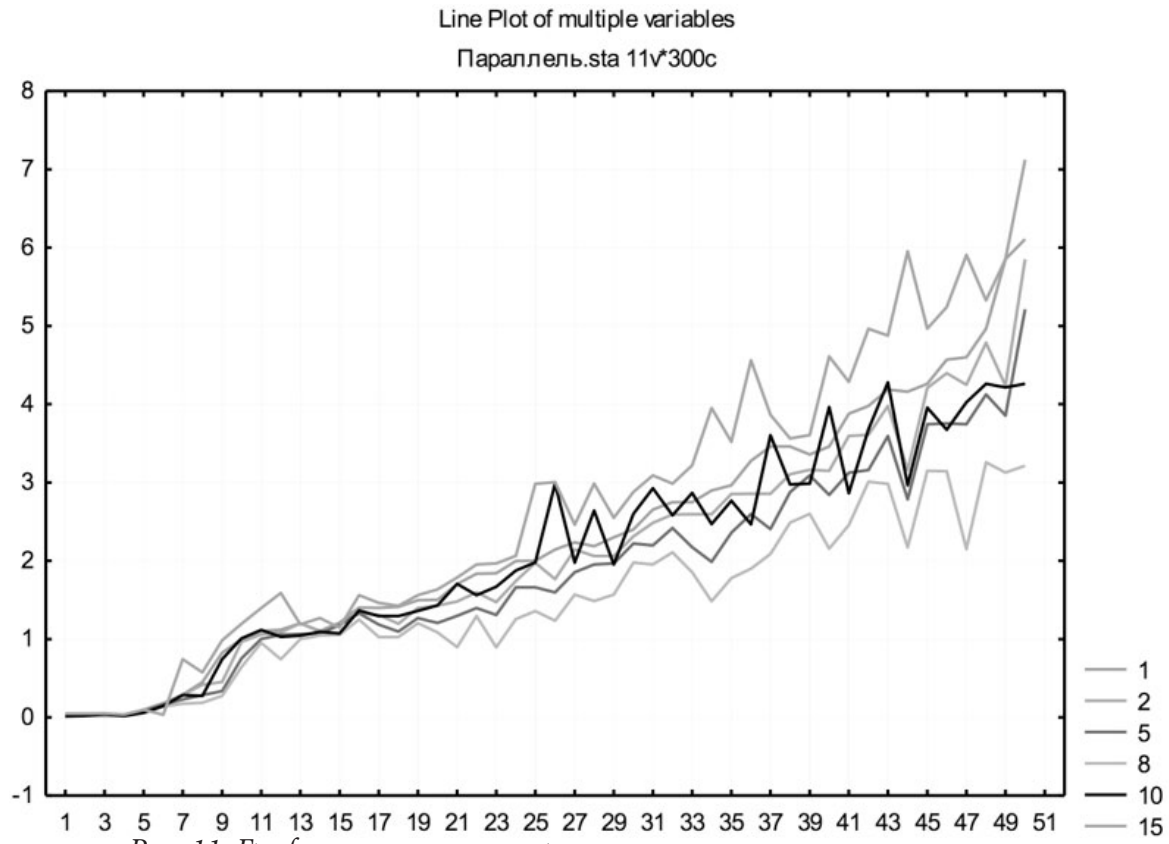

Рис. 11. График зависимости времени вычисления от числа потоков в параллельном релевантном выводе

T-tests; Grouping: Потоки (Параллель.sta)
Group 1: 1
Group 2: 8

Variable	Mean 1	Mean 8	t-value	df	p	Valid N 1	Valid N 8	Std.Dev. 1	Std.Dev. 8	F-ratio Variances	p Variances
Время	2,282412	1,490060	3,034365	98	0,003086	50	50	1,570445	0,971105	2,615248	0,001014

Рис. 12. Результат применения t-критерия

сти времени выполнения алгоритма от числа потоков (рис. 11).

По диаграмме размаха видны различия между средними. Можно заключить, что наиболее высокую эффективность по времени дает использование 8 потоков. Продолжение увеличения числа потоков снижает эффективность, увеличивая время выполнения.

Аналогичный вывод можно сделать и по графику зависимости времени выполнения алгоритма от числа потоков. На оси X отмечено число правил в базах знаний, на оси Y – время выполнения в секундах, а оттенки кривых отражают число потоков при параллельной реализации алгоритма.

Для случая параллельного релевантного вывода с 8 потоками и релевантного вывода применен t-критерий (рис. 9), результаты которого говорят о наличии статистически значимых различий между средними ($p \ll 0{,}05$).

В результате экспериментов было также установлено максимальное количество потоков выполнения (threads), дающее прирост в эффективности на компьютере имеющейся конфигурации (Intel core i7 – 870 (2.93 GHz 8 Mb L3 cache) 4 ядра, 8 потоков, 12 GB 1333MHz DDR3). Анализ результирующих таблиц показывает, что оптимальное количество потоков примерно равно 8.

ЗАКЛЮЧЕНИЕ

Зачастую при создании большой базы знаний не учитывается объем ресурсов, необходимый для ее обработки. В результате этого процесс работы с базой знаний, а также само установление гипотезы, требуют большого объема вычислительных ресурсов. Усовершенствование алгоритмов вывода позволяет уменьшить эти затраты.

Представленная программная библиотека демонстрирует не только работоспособность теории LP-структур в целях ускорения обратного вывода, но и оптимизацию, основанную на применении многопоточности. Параллельный релевантный LP-вывод дает достаточно эффективные подтверждаемые экспериментально результаты для обработки баз знаний больших размеров. Время нахождения решения в случае использования многопоточности снижается на 20–25 % в среднем. В данной ситуации речь идет о статистическом показателе, поскольку и при обычном выводе имеется некоторая вероятность случайного достижения лучшего результата «с первого раза».

СПИСОК ЛИТЕРАТУРЫ

1. *Gupta A.* Parallelism in production systems / Anoop Gupta. – Pitman, 1987. – 224 p.

2. *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.

3. *Болотова С. Ю., Махортов С. Д.* Алгоритмы релевантного обратного вывода, основанные на решении продукцион- но-логических уравнений // Искусственный интеллект и принятие решений. – 2011. – No 2. – С. 40–50.

4. *Рихтер Д.* Windows для профессионалов : Программирование для Windows 95 и Windows NT 4 на базе Win32 API / Пер. с англ. – 3-е изд. – М. : «Рус. Редакция», 1997. – 679 с.

5. *Воеводин В. В.* Параллельные вычисления : Учебное пособие для студ. вузов / В. В. Воеводин, Вл. В. Воеводин. – СПб. : БХВ-Петербург, 2002. – 599 с.

6. *Махортов С. Д.* Интегрированная среда логического программирования LPExpert / С. Д. Махортов // Информационные технологии. – 2009. – № 12. – С. 65–66.

7. Официальный сайт StatSoft Russia. – М. : StatSoft Russia, 2013. – (Электронная библиотека). – http:\\www.statsoft.ru. (дата обращения: март 2017 года).

Болотова С. Ю. – канд. физ.-мат. наук, доцент кафедры математического обеспечения ЭВМ, факультет прикладной математики, информатики и механики, Воронежский государственный университет.
Тел.: +7-951-565-07
E-mail: Bolotova.svetlana@gmail.com

Bolotova S. Yu. – Ph.D. (Phys.-Math.), Department of Applied Mathematics, Mechanics and Informatics, Voronezh State University.
Tel.: +7-951-565-07
E-mail: Bolotova.svetlana@gmail.com