УДК 004.272

Б. А. Кулик
Институт проблем машиноведения РАН (ИПМаш РАН, Санкт-Петербург)

Ю. А. Шичкина
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Распараллеливание алгоритмов решения задач дедуктивного и правдоподобного вывода на основе алгебры кортежей

Рассматриваются возможности эффективного распараллеливания вычислительных процессов, связанных с решением задач дедуктивного и правдоподобного выводов на основе алгебры кортежей, в основе которой лежат матричные структуры.

Дедуктивные и пересматриваемые рассуждения, алгебра кортежей, распараллеливание, информационный граф

Традиционно методы логического (дедуктивного и правдоподобного) анализа систем и рассуждений основаны на формальном подходе, при котором трудно использовать методы эффективного распараллеливания операций. Одним из способов представления данных и знаний является алгебра кортежей, структуры которой обладают естественным параллелизмом. Рассмотрим вкратце основные свойства этой алгебры.

Алгебра кортежей (АК) – это математическая система, предназначенная для анализа разнообразной информации, представленной в виде отношений [1]. К отношениям, в частности, относятся графы, таблицы, правила в продукционных системах, семантические сети и т. д. Универсальность отношений обусловлена хотя бы тем, что предикаты и предложения математической логики являются формальным представлением отношений.

Отношения в АК (***АК-объекты***) представлены с помощью четырех типов структур матричного типа. Каждый АК-объект погружен в определенное пространство ***атрибутов***. В ячейках кортежей АК-объектов содержатся подмножества доменов соответствующих атрибутов, эти подмножества называются ***компонентами***.

Подробные сведения об АК представлены в [1]. Имена АК-объектов содержат идентификатор, к нему добавляется заключенная в прямые скобки последовательность имен атрибутов, определяющих ***схему отношения***, в которой задан этот АК-объект. Например, имя $R[XYZ]$ означает, что АК-объект R задан в пространстве $X \times Y \times Z$, при этом $R[XYZ] \subseteq X \times Y \times Z$. АК-объекты, заданные в одной и той же схеме отношения, называются ***однотипными***.

Свойства АК основаны на свойствах декартова произведения множеств. Базовой структурой в АК является ограниченный прямыми скобками кортеж компонент, например $R[XYZ] = [ABC]$, который называется ***С-кортежем*** и является отношением, равным декартову произведению $A \times B \times C$, при этом $A \subseteq X$; $B \subseteq Y$; $C \subseteq Z$. Элементы декартовых произведений множеств в АК называются ***элементарными кортежами***. Объединение однотипных C-кортежей, выраженное в виде матрицы, называется ***С-системой***. Например, $Q[XYZ] = \begin{bmatrix} A_1 & A_2 & A_3 \\ B_1 & B_2 & B_3 \end{bmatrix}$ есть C-система, ее можно преобразовать в обычное отношение (т. е. в множество элементарных кортежей) следующим образом:

$R[XYZ] = (A_1 \times A_2 \times A_3) \cup (B_1 \times B_2 \times B_3)$, при этом $A_1, B_1 \subseteq X$; $A_2, B_2 \subseteq Y$; $A_3, B_3 \subseteq Z$.

Среди компонент АК имеются особые компоненты, называемые ***фиктивными компонентами.*** К ним относятся ***полная компонента*** (обо-

значается «*»), которая равна домену соответствующего (по месту расположения) атрибута, и пустая компонента («∅»). Например, в *C*-кортеже $P[XYZ] = [A * C]$ полная компонента соответствует множеству, равному домену атрибута *Y*.

Помимо *C*-кортежей и *C*-систем в АК определены еще две структуры – *D*-кортежи и *D*-системы. Первая структура является компактным отображением дополнения соответствующего *C*-кортежа. Например, дополнением *C*-кортежа $R[XYZ] = [ABC]$ является *C*-система $\overline{R}[XYZ] = \begin{bmatrix} \overline{A} & * & * \\ * & \overline{B} & * \\ * & * & \overline{C} \end{bmatrix}$, которая называется ***диагональной C-системой***. Сокращенным обозначением этой *C*-системы является *D*-кортеж $\overline{R}[XYZ] =]\,\overline{A}\ \overline{B}\ \overline{C}\,[$, в котором для отличия от *C*-кортежей используются перевернутые прямые скобки. Другими словами, любой *D*-кортеж можно преобразовать в равносильную ему диагональную *C*-систему.

D-система представляется в виде матрицы, ограниченной перевернутыми прямыми скобками. Каждая строка ее есть *D*-кортеж, а сама *D*-система является отношением, равным пересечению содержащихся в ней *D*-кортежей.

В основе алгоритмов АК лежат операции пересечения, объединения и дополнения АК-объектов. Кроме этих операций в АК предусматриваются операции с атрибутами (элиминация атрибута, добавление фиктивного атрибута, переименование атрибута). Операция ***элиминации атрибута*** соответствует операциям с кванторами в исчислении предикатов. Для ее выполнения достаточно удалить соответствующий атрибут из схемы отношения и соответствующий столбец из матричного представления АК-объекта. Операция ***добавление фиктивного атрибута*** (+Atr) соответствует в исчислении предикатов *правилу обобщения*. Алгоритм ее выполнения состоит из двух простых операций: 1) в схему отношения добавляется имя нового атрибута; 2) в матричное представление преобразуемого АК-объекта добавляется на соответствующем месте столбец с фиктивными компонентами. Например, если в *C*-систему $R[XZ] = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ добавляется фиктивный атрибут *Y*, то будет получена *C*-система $R_1[XYZ] = \begin{bmatrix} A & * & B \\ C & * & D \end{bmatrix}$, которая формально является следствием *C*-системы *R*[*XZ*] по правилу обобщения и семантически равносильна ей.

Если АК-объекты имеют разные схемы отношения, то перед выполнением операций с ними они приводятся к единой схеме отношения за счет добавления недостающих фиктивных атрибутов. Операции $\cap$ и $\cup$ с АК-объектами с предварительным добавлением недостающих фиктивных атрибутов называются ***обобщенными операциями*** и обозначаются соответственно $\cap_G$ и $\cup_G$.

Моделирование дедуктивных и пересматриваемых рассуждений в алгебре кортежей. В теоретических основах АК [1] разработаны алгебраические методы решения следующих задач дедуктивного анализа:

1) проверка корректности определенного следствия *B* для заданных посылок A_i;

2) вывод возможных следствий из заданных посылок A_i с учетом семантических ограничений (например, наличие в следствии определенных переменных или их сочетаний, минимизация состава значащих переменных в следствии и т. д.).

Решение таких задач в АК основано не на правилах вывода, оптимальный порядок применения которых заранее трудно предсказать, а на определенных типовых алгоритмах. Формулы классической логики, представляющие посылки и следствия, выражаются в виде АК-объектов, с которыми можно выполнять обобщенные операции и осуществлять проверку обобщенных соотношений равенства ($=_G$) и включения ($\subseteq_G$). Переход к алгебраическому представлению становится понятным, если учесть, что АК-объекты моделируют область истинности логических формул.

Решение первой задачи дедуктивного анализа, т. е. проверки корректности следствия *B* для заданных посылок A_i, состоит из двух этапов: вначале вычисляется обобщенное пересечение заданных посылок, а затем проверяется включение полученного АК-объекта в предполагаемое следствие *B*:

$$(A_1 \cap_G \ldots \cap_G A_n) \subseteq_G B. \qquad (1)$$

Доказано, что (1) справедливо, если и только если *B* является следствием посылок A_i.

Решение второй задачи дедуктивного анализа (вывод возможных следствий) в АК основано на соотношении (1). Из него следует, что следствием посылок A_i будет любой АК-объект B_j, для которого справедливо $A \subseteq B_j$, где $A = A_1 \cap_G \ldots \cap_G A_n$. Это означает, что любое надмножество *A* будет корректным следствием посылок A_i.

В силу того, что число возможных следствий чрезвычайно велико и значительная их часть не представляет интереса, вторая задача дедуктивного анализа практически целесообразна лишь тогда, когда требуется получить следствие, обладающее определенными свойствами. Для его построения в АК разработано несколько методов [2]. Здесь приведем лишь один из них, в котором вычисляются проекции минимального следствия $A = A_1 \cap_G \ldots \cap_G A_n$. Вычислить проекции АК-объекта A наиболее просто, когда A является C-системой. Тогда B_j получаются элиминацией атрибутов из A. Этот метод позволяет формировать следствия с заданным набором атрибутов.

Рассмотрим вкратце методы решения некоторых задач правдоподобного вывода на основе АК. Будем рассматривать лишь ту часть правдоподобных рассуждений, которые включают пересматриваемые рассуждения. Во многих источниках, в частности в [3]–[5], пересматриваемые рассуждения ассоциируются с неклассическими, в частности, с немонотонными логиками. Анализ доводов, на основе которых многие авторы предлагают при моделировании пересматриваемых рассуждений отказаться от классической логики, показал, что так называемые нарушения монотонности логического вывода в практических рассуждениях на самом деле являются следствиями нарушений неявно заданных ограничений [2]. В теоретических основах АК нарушения ограничений, выявляемые с помощью определенных методов, названы ***коллизиями***.

Анализ показывает, что алгоритмы решения задач дедуктивного вывода и пересматриваемых рассуждений [1], [2] состоят из небольшого числа базовых алгоритмов. В основе всех алгоритмов АК лежат многократно применяемые операции алгебры множеств (пересечение, объединение, дополнение) с компонентами. Поэтому в качестве процессорных элементов (ПЭ) в системах параллельной обработки целесообразно использовать вычислительные устройства, работающие с булевыми векторами произвольной размерности, сопоставимой с размерностью доменов разных атрибутов. В этих ПЭ компоненты АК-объектов представлены характеристическими векторами, а в качестве основных операций применимы векторные булевы операции (конъюнкция, дизъюнкция, отрицание). Кроме того, специфика алгоритмов АК требует включить в состав операций с булевыми векторами операцию распознавания сугубо нулевого вектора. Это позволяет не только распознать пустоту пересечений компонент, но и включение одной компоненты в другую. Так, если проверяется включение компоненты R в компоненту Q, которые представлены булевыми векторами $B(R)$ и $B(Q)$, то соотношение $R \subseteq Q$ справедливо, если и только если в результате операции $B(R)$ **and not** $B(Q)$ будет получен нулевой вектор. Одним из возможных вариантов такого вычислительного устройства может быть защищенное патентом изобретение [6].

Алгоритмы дедуктивного и правдоподобного вывода и их распараллеливание. Все алгоритмы АК, включая алгоритмы дедуктивного и правдоподобного вывода, можно разделить по трудоемкости на следующие классы:

1) операции добавления или удаления фиктивных атрибутов;

2) операции добавления кортежей в АК-объект из другого АК-объекта;

3) операции пересечения (или объединения) пар АК-объектов, когда требуется выполнить операцию пересечения (объединения) каждого кортежа одного АК-объекта с каждым кортежем другого.

Операции первых двух классов не относятся к трудоемким, поэтому их анализ и структурирование здесь не рассматриваются. В других случаях (пересечение C-систем, объединение D-систем, преобразование АК-объектов в альтернативный класс) трудоемкость операций существенно возрастает. В основе этих трудоемких операций лежат две операции: пересечение пары C-кортежей и объединение пары D-кортежей. Рассмотрим первую операцию. Пусть даны 2 однотипных (т. е. погруженных в одно пространство атрибутов) C-кортежа $P = [P_1\ P_2\ \ldots\ P_k]$ и $Q = [Q_1\ Q_2\ \ldots\ Q_k]$. Тогда $R = P \cap Q = [P_1 \cap Q_1\ P_2 \cap Q_2\ \ldots\ P_k \cap Q_k]$. Если хотя бы для одного i $P_i \cap Q_i = \varnothing$, то $R = \varnothing$.

Для удобства построения модели алгоритма все его операции были пронумерованы.

Алгоритм пересечения C-кортежей, приведенных к единой схеме отношения:

1. **алг** Intersection C-n-tuples (P, Q, R, k, F);

2. **нач** ввод компонент кортежей P и Q попарно в память процессорных элементов $ПЭ_r$, …, $ПЭ_{r+k-1}$; $P[i]$, $Q[i]$ – логические векторы, соответствующие компонентам P_i , Q_i; k – размерность C-кортежей:

F – флаг (булева переменная), если $F = 1$, то результат операции принимается (непустой C-кортеж), в противном случае нет,

0[*i*] – логический вектор, у которого все разряды нулевые (соответствует пустому пересечению).

F = 1;

3. **нц для** *i* **от** 1 **до** *k*

4. *R*[*i*]:= *P*[*i*] **and** *Q*[*i*];

5. **если** *R*[*i*] = **0**[*i*] **то** *F* = 0

6. **все**

7. **кц**

8. **если** *F* = 0 **то** *R – пусто*

9. **конец**

В любой программе можно выделить 2 типа действий: преобразователи и распознаватели. Преобразователи перерабатывают информацию, а распознаватели определяют последовательность срабатываний преобразователей в процессе работы программы. Первому типу действий соответствуют, например, операторы присваивания. Их левая часть определяет область памяти программы, подвергающейся воздействию преобразователя, а переменные из правой части показывают на необходимые для выполнения данного действия аргументы. Второму типу действий в программе отвечают альтернативные операторы: условные, разного рода переключатели и т. д. Основное их назначение состоит в выборе одной из нескольких возможных альтернатив дальнейшего следования. Между действиями программы устанавливаются отношения определенного типа. Если одним действием в качестве аргументов используются результаты выполнения других действий, то речь идет об информационной связи. Граф $G = (N, E)$ – это множество N, элементы которого называются узлами, и множество пар (в случае неориентированного графа – неупорядоченных) узлов E, элементы которого называются ребрами. Если среди операторов программы принимать во внимание только преобразователи, а в качестве отношения между ними брать отношение информационной зависимости, то в результате будет получен информационный граф $I = (N_i, E_i)$. Узлы данного графа соответствуют операторам-преобразователям. Два узла соединяются ребром, если между какими-нибудь срабатываниями соответствующих операторов теоретически возможна информационная связь. Информационный граф не зависит от входных данных. В нем могут быть «лишние» ребра, которые не реализуются либо при конкретных входных данных, либо совместно с другими ребрами. Информационный граф представляют собой одну из моделей программы [7].

Вариант 1. Непосредственное распараллеливание.

Построим информационный граф данного алгоритма (рис. 1).

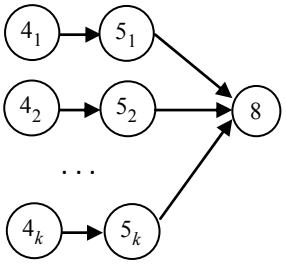

Рис. 1

В соответствии со списочным методом разделения вершин по ярусам найдем [8]:

1-й ярус: $\{4_1, 4_2, \ldots, 4_k\}$;

2-й ярус: $\{5_1, 5_2, \ldots, 5_k\}$;

3-й ярус: $\{8\}$.

Построим по этому распределению операций по процессорам временную диаграмму, полагая, что время выполнения всех операций одинаково и условно равно 1 ед. (рис. 2).

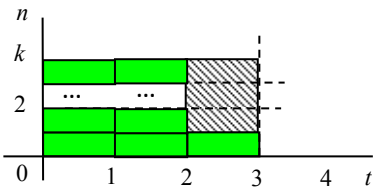

Рис. 2

В результате для алгоритма получаем следующие характеристики:

– число процессоров $n = k$;

– время выполнения $t = 3$;

– количество жестких «пузырей» $p1 = 0$;

– количество ликвидируемых «пузырей» $p2 = k - 1$;

– общее количество «пузырей» $p = p1 + p2 = k - 1$,

где под «пузырями» понимается время простоев процессора; жесткий «пузырь» – время простоя процессора, который не может быть занят другой работой; ликвидируемый «пузырь» – время простоя процессора до окончания работы всего вычислительного комплекса над данным алгоритмом. При удачном планировщике ликвидируемые «пузыри» убираются. В идеальном алгоритме «пузырей» быть не должно, но на практике такая ситуация возможна только при последовательной реализации алгоритма.

Полученные характеристики алгоритма показывают не только хорошую распараллеливаемость алгоритма, но и подтверждают предположения ав-

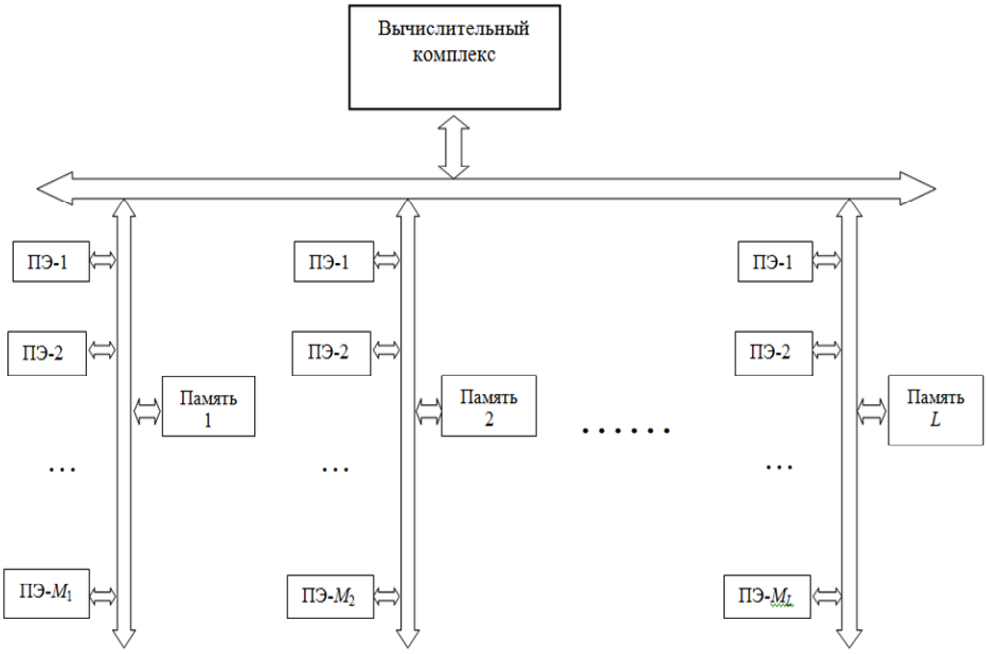

Рис. 3

торов проекта, что параллельную реализацию можно осуществлять на гибридной архитектуре, предложенной в [9] и сочетающей в себе особенности систем с MPP- и SMP-архитектурой (рис. 3).

Пересечение логических векторов $P[i]$, $Q[i]$, соответствующих компонентам P_i , Q_i, и проверка их на пустоту могут осуществляться с помощью процессорных элементов ПЭ-1, ПЭ-2, …, ПЭ-k.

Вариант 2. Блочное распараллеливание.

Чтобы избежать пропорционального длине логических векторов $P[i]$, $Q[i]$ роста объема межпроцессорных передач, можно изначально разбить весь C-кортеж не на отдельные логические векторы $P[i]$, $Q[i]$, а на блоки векторов. Пусть количество векторов в одном блоке равно n_b.

Тогда информационный граф и временная диаграмма останутся прежними, но характеристики поменяются:

число процессоров:

n' = (n – остаток_от_деления (k/n_b))/n_b + 1 = = округление_вниз (k/n_b) + 1;

время выполнения $t = 3n_b$;

количество жестких «пузырей»: $p1 = 0$;

количество ликвидируемых «пузырей» $p2 = n' - 1$;

общее количество «пузырей» $p = n' - 1$.

Разбиение C-кортежей на блоки позволит уменьшить объем вычислительных ресурсов в количество раз, равное размеру блоков, но это приведет к увеличению времени вычислений в такое же количество раз. Межпроцессорных передач в данном алгоритме практически нет, поэтому можно считать, что они не влияют на скорость выполнения алгоритма и не надо применять методы оптимизации по коммуникациям.

Алгоритм объединения D-кортежей, приведенных к единой схеме отношения:

алг Union D-n-tuples (P, Q, R, k, F);

нач ввод компонент кортежей P и Q попарно в память процессорных элементов ПЭ$_r$, …, ПЭ$_{r+k-1}$;

$P[i]$, $Q[i]$ – логические векторы, соответствующие компонентам P_i , Q_i;

k – размерность D-кортежей:

F – флаг (булева переменная), если F = 1, то результат операции принимается (неполный D-кортеж), в противном случае нет,

0[i] – логический вектор, у которого все разряды нулевые (соответствует пустому пересечению).

F = 1;

нц для i **от** 1 **до** k

$R[i]$:= $P[i]$ **or** $Q[i]$;

если not $R[i]$ = **0**[i] **то** F = 0

все

кц

конец

При последовательном выполнении этих алгоритмов цикл можно остановить при первом же нулевом результате операции **and** (для первого алгоритма) или **or** (для второго), однако для обеспечения параллельности этим возможным сокращением вычислений можно пренебречь.

Алгоритмы для вычисления пересечения *C*-кортежей и объединения *D*-кортежей будем рассматривать как процедуры. Они практически совпадают, разница лишь в том, что в цикле первого алгоритма используется логическая операция **and**, а в цикле второго – **or**. Кроме того, с нулевым логическим вектором **0**[*i*] во втором алгоритме сравнивается не результат вычисления *R*[*i*], а его отрицание.

С учетом сходства этих и производных от них алгоритмов будем далее рассматривать и анализировать только операции с *C*-кортежами, т. е. алгоритмы пересечения *C*-кортежа с *C*-системой и *C*-системы с *C*-системой.

Для выполнения операции пересечения *C*-кортежа *P* с *C*-системой *Q* (при условии, что они приведены за счет добавления фиктивных атрибутов к единой схеме отношения) необходимо найти пересечение *C*-кортежа *P* с каждым *C*-кортежем *C*-системы *Q*, при этом в полученную *C*-систему включаются только непустые *C*-кортежи.

Алгоритм пересечения C-кортежа P с C-системой Q:

1. **алг** Intersection C-n-tuple with C-n-system (*P*, *Q*, *W*, *k*, *F*);
2. **нач** пусть все *C*-кортежи содержат *k* компонент, а *C*-система *Q* – *m* *C*-кортежей.

W – *C*-система, полученная в результате операции, которая в данном алгоритме рассматривается как объединение *C*-кортежей. Поскольку эти кортежи имеют одну и ту же схему отношения, то они легко преобразуются в матричное представление. Тогда, используя процедуру Intersection C-n-tuples (*P*, *Q*, *R*, *k*, *F*), нужно выполнить следующий алгоритм.

3. *W*:= ∅;
4. **нц для** *j* **от** 1 **до** *m*
5. **Intersection C-n-tuples (P, Q[j], R[j], k, F)**;
6. **если** *F* = 1 то *W*:= *W*∪ *R*[*j*]
7. **все**
8. **Кц**
9. *W*:= *W*
10. **конец**

Операция 9 введена специально для сбора кортежей в *C*-систему. Операция 9 необходима только для высокопроизводительных систем с распределенной памятью, например кластеров. Для систем с общей памятью операция 9 не нужна.

Информационный граф алгоритма *пересечения C-кортежа с C-системой* отличается от информационного графа алгоритма *пересечения C-кортежей* только количеством процессоров. Но алгоритм *пересечения C-кортежа с C-системой* использует алгоритм *пересечения C-кортежей*. Фактически, каждая 5_i-операция – это алгоритм 1 на *n* процессорах, где $n = k$ или n'. Поэтому информационный граф алгоритма *пересечения C-кортежа с C-системой* с учетом информационного графа алгоритма *пересечения C-кортежей* (см. рис. 1) изменится (рис. 4).

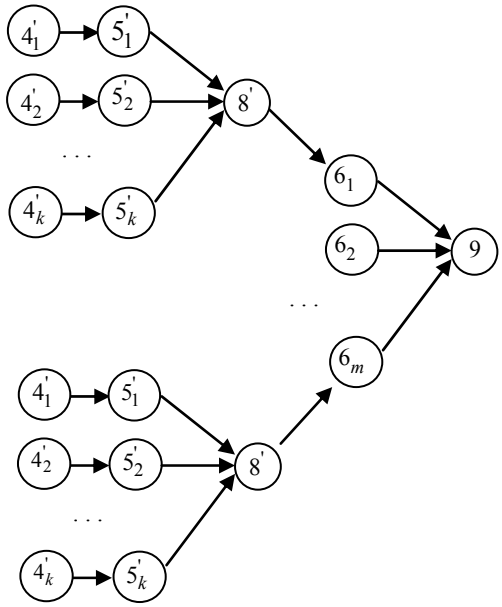

Рис. 4

Операции l_i' – операции, перенесенные из алгоритма *пересечения C-кортежей*.

Не применяя списочный метод оптимизации алгоритма по ширине, получим для алгоритма *пересечения C-кортежа с C-системой* следующие характеристики:

число процессоров $n = km$;

время выполнения $t = 5$;

количество жестких «пузырей» $p1 = 0$;

количество ликвидируемых «пузырей» $p2 = (k-1)m + m - 1 = km - 1$;

общее количество «пузырей»: $p = p1 + p2 = km - 1$.

Соответственно, если вместо разбиения *C*-системы на *k* кортежей применить блочное разбиение, то *k* во всех формулах поменяется на n'.

Для реализации этого алгоритма также может быть использована гибридная архитектура (см. рис. 3).

При выполнении этого алгоритма максимально эффективное распараллеливание достигается, когда одновременно используется $k \times m$ процессорных элементов.

Следующий алгоритм из этой серии также основан на процедуре Intersection C-n-tuples (*P*, *Q*, *R*, *k*, *F*).

Алгоритм пересечения C-системы P с C-системой Q:

1. **алг** Intersection C-n-system with C-n-system (P, Q, W, k, F);

2. **нач** пусть все C-кортежи содержат k компонент, C-система P содержит n C-кортежей, а C-система Q – m C-кортежей.

3. $W := \varnothing$;

4. **нц для** i **от** 1 до n

5. **нц для** j **от** 1 до m

6. **Intersection C-n-tuples (*P*[*i*], *Q*[*j*], *R*[*i*, *j*], *k*, *F*)**;

7. **если** $F = 1$ то $W := W \cup R[i, j]$

8. **все**

9. **кц**

10. **кц**

11. $W := W$

12. **конец**

Информационный граф данного алгоритма аналогичен по структуре информационному графу алгоритма *пересечения C-кортежа P с C-системой Q*, а сам алгоритм имеет следующие характеристики:

– число процессоров $n = kmn$;

– время выполнения $t = 7$;

– количество жестких «пузырей» $p1 = 0$;

– количество ликвидируемых «пузырей» $p2 = (km - 1)n + n - 1 = kmn - 1$;

– общее количество «пузырей» $p = p1 + p2 = kmn - 1$.

Соответственно, если вместо разбиения C-системы на k кортежей применить блочное разбиение, то k во всех формулах поменяется на n'.

Для реализации этого алгоритма также может быть использована гибридная архитектура (см. рис. 3).

Операции l_i' – операции, перенесенные из алгоритма *пересечения C-кортежей*, операции l – операции, перенесенные из алгоритма *пересечения C-кортежа с C-системой*, операции l – операции алгоритма *пересечения C-систем*.

Проведенные исследования позволили подтвердить предположения авторов о наличии в алгоритмах дедуктивного и правдоподобного вывода достаточно большого запаса естественного параллелизма. Также анализ полученных моделей параллельных алгоритмов позволяет сделать вывод, что при выполнении алгоритма пересечения C-систем максимально эффективное распараллеливание достигается, когда одновременно используется $k \times n \times m$ процессорных элементов. Все расчеты были подтверждены с помощью специализированного программного обеспечения [9].

Работа выполнена при финансовой поддержке РФФИ (проекты №№ 14-07-00256-а, 14-07-00257-а).

СПИСОК ЛИТЕРАТУРЫ

1. Кулик Б. А., Зуенко А. А., Фридман А. Я. Алгебраический подход к интеллектуальной обработке данных и знаний. СПб.: Изд-во Политехн. ун-та, 2010. 235 с.

2. Кулик Б. А., Зуенко А. А., Фридман А. Я. Дедуктивные и пересматриваемые рассуждения на основе единого алгебраического подхода // Искусственный интеллект и принятие решений. 2013. Вып. 4. С. 95–105.

3. Рассел С., Норвиг П. Искусственный интеллект: современный подход. 2-е изд. / пер. с англ.; ред. К. А. Птицына. М.: Изд. дом «Вильямс», 2006. 1408 с.

4. Достоверный и правдоподобный вывод в интеллектуальных системах / В. Н. Вагин, Е. Ю. Головина, А. А. Загорянская, М. В. Фомина; под ред. В. Н. Вагина, Д. А. Поспелова. 2-е изд., доп. и испр. М.: ФИЗМАТЛИТ, 2008. 712 с.

5. Логический подход к искусственному интеллекту: от классической логики к логическому программированию / А. Тейз, П. Грибомон, Ж. Луи и др. М.: Мир, 1990.

6. Пат. 2028664 РФ. Устройство для параллельной обработки данных / Б. А. Кулик, Л. Е. Кулик, В. Ф. Федоров. Опубл. 1995. Бюл. № 4.

7. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. СПб.: БХВ-Петербург, 2004.

8. Шичкина Ю. А. Применение списков следования для оптимизации информационного графа по высоте // Системы. Методы. Технологии. 2011. № 1 (9). С. 68–77.

9. Кулик Б. А., Курбанов В. Г., Фридман А. Я. Параллельная обработка данных и знаний методами алгебры кортежей // Тр. СПИИРАН. 2014. Вып. 36. С. 168–179.

B. A. Kulik
Institute of Problems in Mechanical Engineering of the Russian Academy of Sciences (IPME RAS)
Yu. A. Shichkina
Saint Petersburg Electrotechnical University «LETI»

PARALLELIZATION ALGORITHMS FOR SOLVING PROBLEMS OF DEDUCTIVE AND A BELIEVABLE CONCLUSION ON THE BASIS OF THE N- TUPLE ALGEBRA

The possibility of efficient parallelization of computing processes related to problem solving deductive and believable conclusions on the basis of cortege algebra, which is based on the matrix structure.

Deductive and reviewed reasoning, the algebra of tuples, paralleling, information graph