ВОССТАНОВЛЕНИЕ ЛИНЕЙНЫХ ИНДЕКСНЫХ ВЫРАЖЕНИЙ ДЛЯ СВЕДЕНИЯ ПРОГРАММ К ЛИНЕЙНОМУ КЛАССУ

Д.В. Хмелёв

Содержание

Аннотация. Рассматривается класс нелинейных программ, нелинейность которых вызвана оптимизацией кода программы под последовательную ЭВМ или принципиальными ограничениями используемого процедурного языка. Выделены условия, которые необходимо наложить на текст программы из рассматриваемого класса для того, чтобы можно было свести последнюю к программе линейного класса.

1. Введение

Опыт практической эксплуатации параллельных вычислительных систем показал, что одной из самых сложных проблем стала адаптация программного обеспечения под архитектуру конкретной ЭВМ. Появился целый спектр совершенно новых задач: нахождение потенциального параллелизма, преобразование структур данных под многоуровневую память, нахождение “узких мест” и ряд других. Очень часто для

Дата 10 июня 1997 года.

```
      subroutine S1(P,N)
      dimension P(1)
      ii = 1
      do i = 1, N
      do j = 1, i
      P(ii) = 1/(i+j-1)
      ii=ii+1
      end do
      end do
      return
      end
```

```
      subroutine S2(P,N)
      dimension P(1)
C     ii = 1
      do i = 1, N
      do j = 1, i
      P(j+(i-1)*i/2) = 1/(i+j-1)
C     ii=ii+1
      end do
      end do
      return
      end
```

```
      subroutine S3(P,N)
      dimension P(1)
      dimension P17(N,N)
C Expanding P to P17
      ii = 1
      do i = 1, N
      do j = 1, i
      P17(j,i) = P(ii)
      ii=ii+1
      end do
      end do
C Source text with P17
      do i = 1, N
      do j = 1, i
      P17(j,i) = 1/(i+j-1)
C     ii=ii+1
      end do
      end do
C Packing P17 to P
      ii = 1
      do i = 1, N
      do j = 1, i
      P(ii) = P17(j,i)
      ii=ii+1
      end do
      end do
      return
      end
```

Рис. 1

решения подобных задач пользуются только исходным текстом программы, проводят его анализ, пытаясь спрогнозировать поведение программы на этапе выполнения и на основании этого знания делают вывод о возможности целевой оптимизации. Проводимый анализ часто называют *статическим анализом*, поскольку никакой информации со стадии исполнения программы не используется.

Важным источником адаптируемых программных систем является огромный арсенал программного кода, накопленный за полвека использования последовательных ЭВМ. Одним из основных достоинств этих программ является их надежность, проверенная десятилетиями эксплуатации. К сожалению, попытки ускорить исполнение программы на суперкомьютере иногда наталкиваются на трудности, связанные с тем, что программный код оптимизирован под последовательную ЭВМ.

Рассмотрим следующую ситуацию: для вычислительной задачи был составлен алгоритм, в котором существенную роль играют действия с двух- и более мерными массивами; при программной реализации алгоритма, чтобы получить более эффективный код, многомерные массивы были вытянуты в векторы; при этом выражения для индексов массивов усложнились и стали, вообще говоря, нелинейно зависеть от параметров циклов и внешних (по отношению к данному фрагменту программы) переменных. Для того, чтобы при вычислении индексных выражений не использовать операции умножения, вводят *индуктивные переменные*, которые изменяются в циклической конструкции только посредством прибавлений константы и присваиваний таким образом, чтобы на соответствующей итерации они совпадали с нужным индексным выражением.

Другая ситуация связана с использованием структуры данных, которая формально не присутствует в языке, например треугольной матрицы или многомерного массива, координатные размерности которого зависят от внешних переменных. В этом случае многомерные массивы также вытягиваются в векторы и происходит преобразование программы, аналогичное только что изложенному.

В результате программа теряет читаемость и те возможности для ускорения, которые присутствовали в исходном алгоритме. Примером такой программы может послужить подпрограмма `S1`, изображенная на рисунке 1. Мы видим, что индексное выражение массива `P` равно индуктивной переменной `ii`, значение которой каким-то образом зависит от параметров цикла. В большинстве случаев (и, в частности, здесь) можно предъявить *индексную функцию*, которая выражает значение переменной `ii` на конкретной итерации цикла через параметры объемлющих циклов и внешние по отношению к данному фрагменту программы переменные. В данном случае `ii` $= i(i-1)/2+j$. Поиск индексной функции в общем случае представляет отдельную сложную проблему, которая не будет затронута в рамках данной работы. В дальнейшем будем считать, что при исследовании текста программы каким-то образом удалось найти индексные функции и, подставив их вместо индексных переменных, получить, вообще говоря, нелинейное индексное выражение, зависящее только от параметров объемлющих циклов и внешних переменных. Проделав такую операцию над подпрограммой `S1`, получаем подпрограмму `S2`.

Мы рассматриваем программу из класса, которому, в частности, принадлежит `S2`. Нам недоступно описание исходного алгоритма, который она реализует. Наши возможности ограничиваются только формальным анализом текста программы. Логично предположить, что мы оказались именно в той ситуации, которую описали выше и нелинейность индексных выражений вызвана вытягиванием многомерного массива в вектор. Наша задача состоит в том, чтобы восстановить исходную “естественную” запись программы.

В этой статье исследуются условия, при выполнении которых последняя проблема разрешима в случае, когда “упаковывается” треугольная матрица. Мы найдем некоторые естественные достаточные условия того, что линейные индексные выражения могут быть восстановлены исходя только из нелинейного индексного выражения. Оказывается, в общем случае возможно бесконечное число таких восстановлений. Для выделения из этих многих вариантов “настоящего” требуется использовать информацию о структуре объемлющих циклов. Будут найдены достаточные условия конечности числа вариантов восстановления. Самым удивительным результатом работы является теорема 7, которая говорит, что если индексное выражение зависит только от параметров циклов и в пространстве итераций существует вершина, положение которой не зависит от внешних параметров, то существует не более одного варианта восстановления линейных индексных выражений. Например, подпрограмма `S2` удовлетворяет условию этой теоремы.

Пользуясь техникой, изложенной в статье, можно найти этот единственный пригодный вариант восстановленных линейных индексных выражений в подпрограмме `S2`. Проиллюстрируем теперь способ представления исходной записи алгоритма. Преобразуем массив `P` в *полноразмерный* массив `P17`, который состоит из нижнего треугольника с диагональю, содержащего элементы `P`, и верхнего треугольника без диагонали, содержащего неиспользуемые элементы. Таким образом, получаем процедуру `S3`, которая эквивалентна исходной в смысле результатов работы. Индексные выражения второго гнезда циклов, который выполняет вычисления, а, следовательно, наиболее интересен, принимают стандартную линейную форму. Появляется наглядность: становится ясно, что подпрограмма `S3` заполняет упакованный треугольный массив `P` элементами известной матрицы Гильберта порядка `N`.

Единственный неприятный момент состоит в том, что подпрограмма `S3` запрашивает в три раза больше памяти, чем необходимо: вводится новый массив удвоенного размера, но остается и старый. Это результат чересчур прямолинейного восстановления естественной формы записи исходного алгоритма. Если, например, расширить семантику языка Фортран и ввести новый тип “треугольный массив”, то вообще никакой дополнительной памяти не потребуется (но зато потребуется изменить компилятор). Проблему памяти можно решать другими методами: самое главное, что фрагмент программы приобрел естественный вид и возможен адекватный статический анализ этого фрагмента.

Данная работа примыкает к задаче отображения проблем вычислительной математики на архитектуру вычислительных систем, которая рассматривалась в книгах [1] и [2]. Проблематика статьи возникла из желания расширить класс программ, для которых возможен эффективный статический анализ.

2. Модель входного фрагмента программы

В п. 2.1 формально определен линейный класс программ и множество фрагментов, которые мы будем исследовать. П. 2.2 посвящен полному описанию той информации о фрагменте, которую можно получить пользуясь только текстом программы. П. 2.3 посвящен описанию используемого математического аппарата и доказательству важной технической леммы. Далее всюду используются стандартные обозначения: $\mathbf{Z}$ — кольцо целых чисел, $\mathbf{R}$ — поле действительных чисел.

2.1. **Основные определения.** Наиболее распространенным языком программирования для суперЭВМ является Фортран. Согласно установившейся традиции, мы будем изучать фрагменты программ на языке Фортран, хотя все полученные результаты, с несущественными изменениями формулировок, применимы к любому процедурному языку.

Будем говорить, что выражение, содержащееся в операторе S фрагмента, имеет *стандартную линейную форму*, если оно имеет вид: $f_1x_1 + \cdots + f_nx_n + f_0 + f_{n+1}z_1 + \cdots + f_{n+p}z_p$, где $x_1, \ldots, x_n$ — параметры объемлющих оператор S циклов, предположим, что их имеется ровно n; $f_0, \ldots, f_{n+p} \in \mathbf{Z}$; $z_1, \ldots, z_p$ — целочисленные переменные, не изменяемые в данном фрагменте (такие переменные будем называть внешними для данного фрагмента), предположим, что их имеется ровно p. Все операторы, осуществляющие выбор последующего действия в зависимости от истинности некоторого условия, будем называть альтернативными операторами. Тело цикла при каком-то одном значении управляющего параметра назовем итерацией цикла (например, заголовок “`DO 1 i=1,N`” задает N итераций). Будем предполагать, что проверка значения управляющего параметра происходит перед выполнением итерации (так, как принято в Фортране-77).

Определение 1. Фрагмент программы принадлежит *линейному классу* при выполнении следующих условий:

1) он состоит только из операторов присваивания, операторов цикла `DO`, альтернативных операторов (`IF` и пр.) и операторов безусловного перехода, причем все циклы фрагмента заданы только операторами `DO`;

2) все индексные выражения всех обращений к массивам, условия альтернативных операторов, начальные и конечные значения параметров всех циклов имеют стандартную линейную форму;

3) шаг изменения параметров `DO`-циклов равен единице.

Линейный класс программ занимает среди программного обеспечения примерно такое же место, как матрицы в конечномерном анализе, линейные дифференциальные уравнения в теории обыкновенных дифференциальных уравнений, численные методы линейной алгебры в вычислительной математике и т.п. В [1] показано, как для конкретного фрагмента, принадлежащего линейному классу, можно эффективно построить исчерпывающее описание его графа алгоритма с помощью системы функций, линейных как по параметрам цикла, так и по внешним переменным. В [3] доказан фундаментальный факт: для любой программы линейного класса набор таких функций *конечен* и не зависит от значений внешних переменных.

Будем рассматривать проблему преобразования к линейному классу фрагментов, для которых частично не выполнено условие 2. Этот класс будем называть Virtual Dimension или, сокращенно, VD.

Определение 2. Фрагмент программы принадлежит классу VD, если он удовлетворяет условиям 1), 3), и 2), за тем исключением, что индексные выражения некоторых обращений к массивам имеют вид многочлена от параметров объемлющих циклов и внешних переменных.

Не приходится рассчитывать, что любая программа имеет вид, который требуется в последнем определении, но подавляющее большинство программ можно привести к нужному виду, причем привести, разумеется, с гарантией сохранения результатов вычислений. За более подробной информацией читатель может обратиться к статье [4].

2.2. **Модель адресного обращения.** Рассмотрим фрагмент программы класса VD. Множество Ω векторов $\mathbf{z} = (z_1, \ldots, z_p)^T$ будем называть *пространством внешних переменных*. Будем считать, что Ω задается набором линейных неравенств.

Поскольку различие линейным и VD классами состоит только в форме обращений к массивам, нас прежде всего должна интересовать структура конкретного обращения к массиву. Рассмотрим оператор S нашего фрагмента, в котором находится обращение O к массиву, название которого обозначим через P. Вследствие определения фрагментов класса VD, при фиксированном $\mathbf{z}$ множество всех допустимых комбинаций параметров циклов $\mathbf{x} = (x_1, \ldots, x_n)^T$ принимает значения из многогранника $\Omega^I(\mathbf{z})$, который будем называть *пространством итераций*. Из определения класса VD вытекает, что многогранник $\Omega^I(\mathbf{z})$ описывается с помощью параметризованной системы линейных неравенств $\mathbf{Ax} \leq \mathbf{b}(\mathbf{z})$, где каждая компонента $\mathbf{b}(\mathbf{z})$ имеет стандартную линейную форму. Будем говорить, что выражение для индекса массива P по k-той координате имеет *стандартную полиномиальную форму*, если оно записывается в виде многочлена $g(\mathbf{x}, \mathbf{z})$ многих переменных над полем рациональных чисел. Предположим, что для любого $\mathbf{z} \in \Omega$ множество $\Omega^I(\mathbf{z})$ непусто и ограничено.

Определение 3. Будем называть *адресным обращением* набор

$$\big\{S,\ O,\ P,\ k,\ \mathbf{z},\ \Omega,\ \mathbf{x},\ \mathbf{A},\ \mathbf{b}(\mathbf{z}),\ g(\mathbf{x},\mathbf{z})\big\} \tag{2.1}$$

удовлетворяющий приведенным выше условиям.

Например, в подпрограмме $S2$ адресное обращение единственно и его компоненты определяются следующим образом:

$$S, O, P = \text{“P”}, k = 1, \mathbf{z} = (N), \Omega = \{N \mid N \geq 1\},$$

$$\mathbf{x} = \begin{pmatrix} i \\ j \end{pmatrix}, \mathbf{b} = \begin{pmatrix} -1 \\ N \\ -1 \\ 0 \end{pmatrix}, \mathbf{A} = \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ -1 & 1 \end{pmatrix}, g(\mathbf{x},\mathbf{z}) = \frac{x_1(x_1-1)}{2} + x_2.$$

2.3. **Параметрическое линейное программирование.** Восстановленные линейные индексы многомерного массива обязаны удовлетворять некоторым линейным неравенствам, которые должны быть выполнены при всех допустимых комбинациях параметров объемлющих циклов. Для проверки неравенств необходимо находить минимум линейного функционала на пространстве итераций, которое зависит от внешних переменных. В этом пункте мы развиваем математический аппарат, необходимый для решения этой задачи.

Построим алгоритм нахождения параметрического минимума стандартной линейной формы $f_1x_1 + \dots + f_nx_n + f_0 + f_{n+1}z_1 + \dots + f_{n+p}z_p = (\mathbf{f}_x, \mathbf{x}) + (\mathbf{f}_z, \mathbf{z}) + f_0$, где $(\cdot,\cdot)$ обозначает скалярное произведение в пространствах одинаковой размерности, $\mathbf{f}_x = (f_1, \dots, f_n)^T$, $\mathbf{f}_z = (f_{n+1}, \dots, f_{n+p})^T$, индекс T обозначает транспонирование:

$$\min_{\mathbf{Ax} \leq \mathbf{b}(\mathbf{z})} [(\mathbf{f}_x, \mathbf{x}) + (\mathbf{f}_z, \mathbf{z}) + f_0] = (\mathbf{f}_z, \mathbf{z}) + f_0 + \min_{\mathbf{Ax} \leq \mathbf{b}(\mathbf{z})} (\mathbf{f}_x, \mathbf{x}).$$

Обозначим $\mathbf{c} = \mathbf{f}_x$. Выше мы потребовали, чтобы множество Ω задавалось набором линейных неравенств. Поэтому если подразумевается $\mathbf{z} \in \mathbf{Z}^p$ (соответственно, $\mathbf{z} \in \mathbf{R}^p$), то под множеством Ω мы понимаем все целочисленные (соответственно, все действительные) решения этих неравенств. Рассмотрим следующую задачу *параметрического линейного программирования* на линейном пространстве $\mathbf{L} = \mathbf{Z}, \mathbf{R}$:

(2.2)

$$\mathcal{P}\mathbf{L}(\mathbf{z}, \Omega) : \min_{\mathbf{Ax} \leq \mathbf{b}(\mathbf{z})} (\mathbf{c}, \mathbf{x}), \text{ где } \mathbf{A} = (a_{ij})_{m \times n} \in \mathbf{Z}^{m \times n},\ \mathbf{z} \in \Omega \subset \mathbf{L}^p,\ \mathbf{c} \in \mathbf{Z}^n,\ \mathbf{x} \in \mathbf{L}^n,$$

$$\mathbf{b}(\mathbf{z}) = (b_1(\mathbf{z}), \dots, b_m(\mathbf{z}))^T,\ b_i(\mathbf{z}) = (\mathbf{b}_i, \mathbf{z}) + b_{i0},\ \mathbf{b}_i \in \mathbf{Z}^p,\ b_{i0} \in \mathbf{Z} \text{ для } i = \overline{1, m}.$$

Мы считаем, что множество Ω обладает свойством $\forall \mathbf{z} \in \Omega$ *допустимое множество* $M(\mathbf{A}, \mathbf{b}(\mathbf{z})) = \{\mathbf{x} \in \mathbf{L}^n \mid \mathbf{Ax} \leq \mathbf{b}(\mathbf{z})\}$ непусто и ограничено:

это является вполне естественным условием конечности времени исполнения и нетривиальности фрагмента программы. Для выделения множества Ω исходя из системы неравенств $\mathbf{Ax} \leq \mathbf{b}(\mathbf{z})$ в случае $\mathbf{L} = \mathbf{R}$ можно воспользоваться методом Фурье-Моцкина, описанным в [5]. Если пространство $\mathbf{L}$ целое ($\mathbf{L} = \mathbf{Z}$), матрица $\mathbf{A}$ и вектор-функция $\mathbf{b}(\mathbf{z})$ совпадают с матрицей и вектор-функцией адресного обращения (2.1), то $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ совпадает с $\Omega^I(\mathbf{z})$: $M(\mathbf{A}, \mathbf{b}(\mathbf{z})) = \Omega^I(\mathbf{z})$.

Вектор $\mathbf{x}^*(\mathbf{z})$ является решением, если при фиксированном $\mathbf{z}$ для любых допустимых $\mathbf{x}$ $(\mathbf{c}, \mathbf{x}^*(\mathbf{z})) \leq (\mathbf{c}, \mathbf{x})$. *Значением* задачи $\mathcal{P}\mathbf{L}$ называется функция $\varphi(\mathbf{z}) = (\mathbf{c}, \mathbf{x}^*(\mathbf{z}))$.

Рассмотрим двойственную к (2.2) задачу линейного программирования

$$\mathcal{D}\mathbf{L}(\mathbf{z}, \Omega) : \max_{\mathbf{A}^T\mathbf{y}=\mathbf{c};\, \mathbf{y}\leq 0} (\mathbf{b}(\mathbf{z}), \mathbf{y}), \text{ где } \mathbf{y} \in \mathbf{L}^m. \tag{2.3}$$

Вектор $\mathbf{y}^*(\mathbf{z})$ является решением, если при фиксированном $\mathbf{z}$ для любых допустимых $\mathbf{y}$: $(\mathbf{b}(\mathbf{z}), \mathbf{y}^*(\mathbf{z})) \geq (\mathbf{b}(\mathbf{z}), \mathbf{y})$. *Значением* задачи $\mathcal{D}\mathbf{L}(\mathbf{z}, \Omega)$ называется функция $(\mathbf{b}(\mathbf{z}), \mathbf{y}^*(\mathbf{z}))$.

Из непустоты и ограниченности $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$, следует, что значение $\mathcal{D}\mathbf{R}(\mathbf{z}, \Omega)$ существует и совпадает со значением $\mathcal{P}\mathbf{R}(\mathbf{z}, \Omega)$ (см. [6]). Многогранник ограничений задачи $\mathcal{D}\mathbf{R}(\mathbf{z}, \Omega)$ также непуст и ограничен, а, следовательно, имеет конечное число w вершин $\mathbf{y}^1, \ldots, \mathbf{y}^w$. Поскольку значение задачи $\mathcal{D}\mathbf{R}(\mathbf{z}, \Omega)$ совпадает со значением функционала $(\mathbf{b}(\mathbf{z}), \mathbf{y})$ в некоторой вершине допустимого множества задачи $\mathcal{D}\mathbf{R}(\mathbf{z}, \Omega)$, справедлива формула:

$$\varphi(\mathbf{z}) = \max_{k=\overline{1,w}} (\mathbf{b}(\mathbf{z}), \mathbf{y}^k). \tag{2.4}$$

Лемма 1. *Для полиэдра $\Omega \subset \mathbf{R}^p$ такого, что $\forall \mathbf{z} \in \Omega$ допустимое множество $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ непусто и ограничено, существует конечное, не зависящее от значений внешних переменных, разбиение на полиэдры, такое что на каждом из них координаты вершин $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ являются линейными формами с рациональными коэффициентами от $\mathbf{z}$.*

Доказательство. Наше доказательство состоит в построении нужного разбиения. Введем необходимые обозначения. Пусть задано множество $v \subset \{1, \ldots, m\}$. Упорядочим элементы v по возрастанию: $v = \{v_1 < v_2 < \ldots < v_k\}$. Для произвольного вектора a мы понимаем под a_v вектор из k компонент, имеющий следующий вид: $a_v = (a_{v_1}, a_{v_2}, \ldots, a_{v_k})^T$. Обозначим через $\mathbf{A}_v$ матрицу из строк $v_1, \ldots, v_k$ матрицы $\mathbf{A}$. Рассмотрим задачи параметрического линейного программирования для $\mathbf{c} = -(\mathbf{A}_{\{i\}})^T$.

$$\mathcal{P}\mathbf{R}_i(\mathbf{z}, \Omega) : \min_{\mathbf{Ax}\leq\mathbf{b}(\mathbf{z})} (\mathbf{c}, \mathbf{x}).$$

Двойственная задача имеет вид

$$\mathcal{D}\mathbf{R}_i(\mathbf{z},\Omega): \max_{\mathbf{A}^T\mathbf{y}=\mathbf{c};\,\mathbf{y}\le 0}(\mathbf{b}(\mathbf{z}),\mathbf{y}).$$

Базисом называется всякая невырожденная матрица, составленная из n строк матрицы $\mathbf{A}$. Пусть номера строк задаются множеством v. *Допустимым базисом* задачи $\mathcal{D}\mathbf{R}_i$ называется такая матрица $B = \mathbf{A}_v$, что $(B^T)^{-1}\mathbf{c} \le 0$. Рассмотрим точку $\mathbf{x}(\mathbf{z}) = B^{-1}b_v(\mathbf{z})$. Обозначим $\bar{v} = \{1, \ldots, m\} \setminus v$, $N = \mathbf{A}_{\bar{v}}$. Ввиду [7, Глава 2, 1.6, теорема 3] точка $\mathbf{x}(\mathbf{z})$ является вершиной допустимого множества задачи $\mathcal{P}\mathbf{R}_i$ тогда и только тогда, когда выполнена система неравенств $N\mathbf{x}(\mathbf{z}) \le \mathbf{b}_{\bar{v}}(\mathbf{z})$ или $\mathbf{A}_{\bar{v}}(\mathbf{A}_v)^{-1}\mathbf{b}_v(\mathbf{z}) \le \mathbf{b}_{\bar{v}}(\mathbf{z})$. Последняя система определяет полиэдр в пространстве внешних переменных, для которого координаты вершины $\mathbf{x}(\mathbf{z})$ линейно зависят от $\mathbf{z}$. Перебрав все допустимые базисы для задачи $\mathcal{D}\mathbf{R}_i$ и решив соответствующие системы, мы найдем, вообще говоря, неоднородные кусочно-линейные не всюду определенные функции, которые задают координаты вершин полиэдра $M(\mathbf{A},\mathbf{b}(\mathbf{z}))$. Эти вершины лежат на грани $M(\mathbf{A},\mathbf{b}(\mathbf{z}))$, которая лежит в плоскости $(\mathbf{c},\mathbf{x}) = -b_i(\mathbf{z})$ (или $\mathbf{A}_{\{i\}}\mathbf{x} = b_i(\mathbf{z})$); число кусков линейности каждой координатной функции конечно. Носитель каждой функции является объединением конечного числа полиэдров.

Аналогичное построение верно для $i = \overline{1,m}$. Утверждение леммы вытекает из конечности числа носителей функций и конечности числа вершин. $\square$

Значение "непрерывной" задачи $\mathcal{P}\mathbf{R}(\mathbf{b}(\mathbf{z}),\Omega)$ при целом $\mathbf{z}$ всегда меньше либо равно значения "целой" задачи $\mathcal{P}\mathbf{Z}(\mathbf{b}(\mathbf{z}),\Omega)$. Они совпадают, если многогранник ограничений $\mathbf{A}\mathbf{x} \le \mathbf{b}(\mathbf{z})$ является *целочисленным*, то есть все его вершины при целых $\mathbf{z}$ имеют только целочисленные координаты. Такие полиэдры возникли в связи с так называемой "транспортной задачей". Еще основателями теории линейного программирования, был описан класс таких матриц $\mathbf{A}$, что многогранник ограничений $M(\mathbf{A},\mathbf{b})$ является целочисленным при любом целом $\mathbf{b}$.

Заметим, что наш класс матриц включается в этот. Действительно, вектор $\mathbf{b}(\mathbf{z})$ при целочисленном $\mathbf{z}$ является целочисленным, но зависит не от m параметров, а от p, причем совсем не обязательно $p = m$. На практике $p \ll m$, чаще всего $p = 1$, реже $p = 2$. Поэтому теоремы, рассмотренные ниже, дают достаточные условия на целочисленность $M(\mathbf{A},\mathbf{b}(\mathbf{z}))$.

Определение 4. Назовем целочисленную матрицу $\mathbf{A}$ *абсолютно унимодулярной*, если все ее миноры равны либо 0, либо ± 1.

Теорема 1. *Множество $M(\mathbf{A}, \mathbf{b})$ целочисленно при любом целочисленном векторе $\mathbf{b}$ тогда и только тогда, когда матрица $\mathbf{A}$ абсолютно унимодулярна.*

Эта теорема доказана в [8], но более фундаментальное рассмотрение — в [9]. ограничений. При больших m и n проверка унимодулярности может занять много времени, поэтому сформулированы достаточные признаки унимодулярности.

Теорема 2. *Пусть $\mathbf{A}$ — целочисленная матрица из $\{\pm 1, 0\}$ с не более чем двумя ненулевыми элементами в каждом столбце. Для того чтобы $\mathbf{A}$ была абсолютно унимодулярной, необходимо и достаточно, чтобы строки еМожно было разбить на два класса, обладающих следующими свойствами.*

1) Если строки i и k принадлежат одному и тому же классу и для некоторого j имеем $a_{ij} \neq 0$, $a_{kj} \neq 0$, то $a_{ij} = -a_{kj}$.

2) Если строки i и k принадлежат разным классам, и для некоторого j имеем $a_{ij} \neq 0$, $a_{kj} \neq 0$, то $a_{ij} = a_{kj}$.

Доказательство теоремы и алгоритм проверки достаточного условия на унимодулярность приведены в [10, Глава 17]. См. также [11]. Практика показывает, что исключительно много адресных обращений в реальных программах обладают абсолютно унимодулярной матрицей.

3. Восстановление треугольных массивов

3.1. **О хранении элементов массивов в памяти.** Пусть требуется хранить все элементы квадратной матрицы $T(I, J)$, $I, J = \overline{1, q}$ в последовательных ячейках памяти. Самым естественным (и наиболее часто используемым) способом распределения памяти является такой способ, при котором матрица располагается в памяти в лексикографическом порядке индексов или, как иногда говорят, “по строкам”:

$$T(1,1), T(1,2), \ldots, T(1,q), T(2,1), \ldots, T(q,q-1), T(q,q). \tag{3.1}$$

Например, именно так организовано хранение матриц в языке Си. В языке Фортран используется метод, при котором матрица располагается в памяти в антилексикографическом порядке индексов. Пара индексов (I, J) антилексикографически старше пары индексов (I', J'), если $J > J'$. В случае $J = J'$, пара (I, J) старше пары (I', J'), если $I > I'$. При таком способе матрица $T(I, J)$ хранится следующим образом:

$$T(1,1), T(2,1), \ldots, T(q,1), T(1,2), \ldots, T(q-1,q), T(q,q). \tag{3.2}$$

Обозначим через $\mathrm{LOC}_{lex}(T(I, J))$ адрес в памяти элемента $T(I, J)$, хранящегося способом (3.1), а через $\mathrm{LOC}_{antilex}(T(I, J))$, соответственно, способом (3.2). Для произвольных K, $L = \overline{1, q}$

$$\mathrm{LOC}_{lex}(T(K, L)) = \mathrm{LOC}_{antilex}(T(L, K)).$$

В дальнейшем будем считать, что массивы хранятся в памяти лексикографическом порядке индексов, но, записывая обращение к элементу массива языка Фортран, будем заменять порядок индексов на обратный. Именно поэтому мы переставили индексы `i` и `j` в обращении к массиву `P17` в программе `S3` на рисунке1. Смысл этой операции состоит в том, что сохраняется линейная упорядоченность в памяти элементов упакованного массива, которые составляли, в случае с `S2` и `S3`, столбец восстановленного массива. За более подробной информацией можно обратиться к книге [12, 2.2.6].

3.2. **Варианты хранения треугольного массива.** Пусть элементы массива $T(J, I)$ определены не для всех индексов, а только для одного из следующих множеств:

$$A_1 = \left\{(I, J) \mid I = \overline{1, q}, J = \overline{1, I}\right\},\ A_2 = \left\{(I, J) \mid I = \overline{1, q}, J = \overline{1, q - I + 1}\right\},$$
$$A_3 = \left\{(I, J) \mid I = \overline{1, q}, J = \overline{q - I + 1, q}\right\},\ A_4 = \left\{(I, J) \mid I = \overline{1, q}, J = \overline{I, q}\right\}.$$

В каждом из приведенных множеств $q(q + 1)/2$ пар. Предположим, что элементы из T хранятся в одномерном массиве $T'(1 : q(q+1)/2)$ в лексикографическом порядке индексов. Индекс элемента $T(J, I)$ в массиве T' определяется с помощью специальных функций, представленных в таблице.

Таблица

множество индексов	соответствующая функция
A_1,	$F(I, J) = I(I - 1)/2 + J$,
A_2,	$G(I, J) = -I(I - 1)/2 + q(I - 1) + J$,
A_3,	$F'(I, J) = F(I, J - (q - I) + 1)$,
A_4,	$G'(I, J) = G(I, J - I + 1)$.

Рассмотрим адресное обращение (2.1) к массиву $T'(1 : q(q + 1)/2)$. Предположим, что индексные выражения $L(\mathbf{x}, \mathbf{z})$ и $R(\mathbf{x}, \mathbf{z})$ массива T имели стандартную линейную форму, то есть, на итерации $\mathbf{x}$ циклической конструкции производилось обращение к элементу $T(R(\mathbf{x}, \mathbf{z}), L(\mathbf{x}, \mathbf{z}))$. Пара индексных выражений $(L(\mathbf{x}, \mathbf{z}), R(\mathbf{x}, \mathbf{z}))$ принадлежит множеству индексов A_t, где $t \in \{1, 2, 3, 4\}$. На итерации $\mathbf{x}$ обращение к элементу массива T', соответствующему $T(R(\mathbf{x}, \mathbf{z}), L(\mathbf{x}, \mathbf{z}))$, можно записать в виде функции от $R(\mathbf{x}, \mathbf{z})$ и $L(\mathbf{x}, \mathbf{z})$, соответствующей множеству A_t. Из текста программы известна функция $g(\mathbf{x}, \mathbf{z})$, которая получается после подстановки линейных индексных выражений в специальную функцию. Если, например, $t = 3$, то $g(\mathbf{x}, \mathbf{z}) = F'(L(\mathbf{x}, \mathbf{z}), R(\mathbf{x}, \mathbf{z}))$.

Наша общая задача восстановления линейных индексных выражений в данном случае состоит в том, чтобы выделить множества $\Omega_k \subset \Omega$, на которых можно задать стандартные линейные формы $L_k(\mathbf{x}, \mathbf{z})$ и $R_k(\mathbf{x}, \mathbf{z})$, удовлетворяющие следующим условиям

а) $\big(L_k(\mathbf{x}, \mathbf{z}), R_k(\mathbf{x}, \mathbf{z})\big) \in A_t$ для любого $\mathbf{x} \in M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$,

б) функция, соответствующая A_t, от $L_k(\mathbf{x}, \mathbf{z})$ и $R_k(\mathbf{x}, \mathbf{z})$ равна $g(\mathbf{x}, \mathbf{z})$.

Поскольку мы хотим найти представление $g(\mathbf{x}, \mathbf{z})$ в виде композиции специального многочлена второй степени и стандартных линейных форм L и R, то сразу возникает необходимое ограничение на степень g: она не должна превосходить 2.

Посредством линейных замен пары из A_3 и A_4 переводятся в пары из A_1 и A_2 соответственно. Поэтому для $g(\mathbf{x}, \mathbf{z})$ должно быть справедливо одно из представлений:

- $g(\mathbf{x}, \mathbf{z}) = F(L(\mathbf{x}, \mathbf{z}), R(\mathbf{x}, \mathbf{z}))$, $\big(L(\mathbf{x}, \mathbf{z}), R(\mathbf{x}, \mathbf{z})\big) \in A_1$,
- $g(\mathbf{x}, \mathbf{z}) = G(L(\mathbf{x}, \mathbf{z}), R(\mathbf{x}, \mathbf{z}))$, $\big(L(\mathbf{x}, \mathbf{z}), R(\mathbf{x}, \mathbf{z})\big) \in A_2$.

Классификация. Знак коэффициента при квадрате любой переменной вектора $\mathbf{x}$ в многочлене $g(\mathbf{x}, \mathbf{z})$ является инвариантом, однозначно определяющим выбор представления:

1) если коэффициент положителен, то $g = F(\cdot, \cdot)$,

2) если коэффициент отрицателен, то $g = G(\cdot, \cdot)$.

Если у квадратов разных переменных $\mathbf{x}$ различны знаки, то восстановить линейные индексные выражения нельзя. Если квадраты переменных $\mathbf{x}$ отсутствуют, то возможны два случая.

Случай 1. $g(\mathbf{x}, \mathbf{z})$ является линейной неоднородной функцией $\mathbf{x}$ — тогда выбор представления следует согласовывать с другими обращениями к этому массиву; при этом возможны два варианта:

а) $g(\mathbf{x}, \mathbf{z}) = F(a(\mathbf{z}), R(\mathbf{x}, \mathbf{z}))$, $1 \le R(\mathbf{x}, \mathbf{z}) \le a(\mathbf{z})$, или $g(\mathbf{x}, \mathbf{z}) = G(a(\mathbf{z}), R(\mathbf{x}, \mathbf{z}))$, $1 \le R(\mathbf{x}, \mathbf{z}) \le q - a(\mathbf{z}) + 1$, здесь $L(\mathbf{x}, \mathbf{z})$ равно линейной неоднородной функции $a(\mathbf{z})$, это означает, что программа обращается непосредственно к строке $a(\mathbf{z})$ треугольной матрицы;

б) не существует такой константы a, чтобы было выполнено одно из условий предыдущего пункта, это означает, что массив проходится нерегулярно — циклическая конструкция не согласована со структурой многомерного массива; можно попытаться согласовать данное адресное обращение с другими к этому же массиву при помощи расщепления циклов.

Случай 2. В противном случае нельзя представить $g(\mathbf{x}, \mathbf{z})$ в виде композиции одной из функций F или G и линейных форм (таким образом, адресные обращения вроде $T(I * J)$) исключены).

Рассмотрим технику восстановления линейных индексных выражений для адресного обращения с функцией F. Аналогичную технику можно применить для случая с G.

3.3. **Основное разложение индексной функции.** Найдем разложение функции $g(\mathbf{x}, \mathbf{z})$ в предположении, что коэффициент при квадрате любой переменной $g(\mathbf{x}, \mathbf{z})$ больше нуля (это соответствует представлению $g(\mathbf{x}, \mathbf{z}) = F(L(\mathbf{x}, \mathbf{z}), R(\mathbf{x}, \mathbf{z})))$. Введем обозначение $\mathbf{y}^T = (\mathbf{x}^T, \mathbf{z}^T)$.

Определение 5. Назовем функцию

$$Q[\mathbf{y}] = \sum_{i=1}^{n+p} \sum_{j=1}^{n+p} q_{ij} y_i y_j$$

квадратичной формой от вектора $\mathbf{y} \in \mathbf{R}^{n+p}$.

Лемма 2. *Квадратичная форма* $Q[\mathbf{y}] \neq 0$ *разлагается в квадрат линейного функционала от* $\mathbf{y}$ *тогда и только тогда, когда она симметрична с неотрицательными элементами на диагонали и каждые два ненулевых столбца матрицы* Q *пропорциональны.*

Будем у произвольной функции $f(\mathbf{x}, \mathbf{z})$ опускать аргументы $\mathbf{x}$ и $\mathbf{z}$, если речь идет об ее символьной записи. В этом случае под операциями сложения и вычитания, умножения, подстановки следует понимать соответствующие операции с символьными записями с приведением подобных членов. Если $g = F(L, R)$, L и R имеют стандартную линейную форму, то справедливо разложение $2g(\mathbf{x}, \mathbf{z}) = Q[\mathbf{y}] + f(\mathbf{y})$, где $Q[\mathbf{y}]$ удовлетворяет условиям леммы, а $f(\mathbf{y})$ — линейная форма. Пусть $Q[\mathbf{y}] = (\tilde{L}(\mathbf{y}))^2$. Линейную форму $L(\mathbf{y})$ можно определить с точностью до знака и константы: $L(\mathbf{y}) = \pm\tilde{L}(\mathbf{y}) + const$. Поэтому если коэффициенты линейной формы $\tilde{L}(\mathbf{y})$ не целые, то восстановить линейные индексные выражения нельзя.

Вычислим символьно:

$$\tilde{R} = g - \tilde{L}(\tilde{L} - 1)/2.$$

Если у линейной формы $\tilde{R}$ существует нецелый коэффициент, то восстановить линейные индексные выражения нельзя. В дальнейшем будем предполагать, что $\tilde{L}(\mathbf{x}, \mathbf{y})$ и $\tilde{R}(\mathbf{x}, \mathbf{y})$ имеют стандартную линейную форму.

```
do i=p,p+q
do j=r,r+s
A(j+i*(i-1)/2)=i+j
enddo
enddo
```

Рис. 2

3.4. **Эвристические соображения.** Существует ли фрагмент программы класса VD, который имеет бесконечно много вариантов восстановления линейных индексных выражений (для различных внешних параметров)? Как ни странно, такой фрагмент действительно существует. Его код приведен на рисунке 2.

Исследуем этот фрагмент в предположении, что нам заведомо ничего не известно про параметры p, q, r и s. В первую очередь заметим, что для адресации используется специальная функция $F(\cdot,\cdot)$. С помощью пункта 3.3 выделяем $L_0 = \tilde{L} = i$, $R_0 = \tilde{R} = j$. На первый взгляд, в программе обрабатывается прямоугольник размера $(q{+}1)\times(s{+}1)$. Тем более удивительно, что это не единственный вариант обработки упакованного треугольного массива. Зададим

$$L_k = L_{k-1} + 1 = i + k;\ R_k = R_{k-1} - L_{k-1} = j - ki - k(k-1)/2,\ k \in \mathbf{Z}.$$

Из вида F вытекает $F(L_0, R_0) = F(L_k, R_k)$. Программа, изображенная на рисунке 2, обрабатывает фрагмент треугольного массива тогда и только тогда, когда для всех точек пространства итераций выполнено: $L_k(i,j) \geq 1$, $R_k(i,j) \geq 1$, $\big[L_k(i,j) \geq R_k(i,j) \iff (L_k - R_k)(i,j) \geq 0\big]$. Составим систему неравенств на внешние параметры: для $i = \overline{p, p+q}$, $j = \overline{r, r+s}$,

$$\left\{\begin{array}{l} L_k(i,j) \geq 1, \\ R_k(i,j) \geq 1, \\ (L_k - R_k)(i,j) \geq 0 \end{array}\right. \iff \left\{\begin{array}{l} p + k \geq 1, \\ r - k(p+q) - k(k-1)/2 \geq 1, \\ -(r+s) + (k+1)p + k(k+1)/2 \geq 0. \end{array}\right.$$

Найдем нетривиальное решение последней системы.

$$k = t,\ q = t,\ s = t,\ p = 1{+}t^2,\ r = t^3{+}t^2{+}t(t{+}1)/2{+}1, \text{ где } t = 1,\ 2,\ \ldots.$$

При такой параметризации

$$\begin{aligned} & r - k(p+q) - k(k-1)/2 = t^3 + t^2 + t(t+1)/2 + 1 - t(1+t^2) - t^2 - t(t-1)/2 = 1, \\ & -(r+s) + (k+1)p + k(k+1)/2 = -\big(t^3 + t^2 + t(t+1)/2 + 1\big) - t \\ & \qquad + (1+t)(1+t^2) + \frac{t(t+1)}{2} = 0 \end{aligned}$$

и требуемые неравенства выполняются. Следовательно, требуемый пример построен и в классе VD существуют программы, которые могут иметь бесконечно много вариантов восстановления линейных индексных выражений. В нашем примере такими выражениями являются L_t и R_t для $t = 1, 2, \ldots$.

Может показаться, что пример довольно сомнителен: в реальной программе значения p, q, r и s определены заранее и к началу исполнения фрагмента они известны, а следовательно, такого изобилия вариантов может и не быть. Поэтому еще раз подчеркнем, что мы рассматриваем фрагмент отдельно от всей программы и про p, q, r и s заведомо ничего не известно. Такая постановка задачи вполне соответствует духу статического анализа: определить структуру фрагмента, и лишь затем согласовывать ее с остальными структурами. Разобранный пример демонстрирует, что множество программ, которые можно задать с помощью процедурного языка, значительно шире множества реальных программ. Редкий разработчик программного обеспечения закладывает в текст программы бесконечно большое количество вариантов ее поведения в зависимости от значений внешних параметров. Поэтому на практике такое разнообразие не встречается, а, следовательно, встает задача выделения класса таких фрагментов, для которых существует лишь конечное число вариантов их восстановления. Теоремы следующего пункта дают возможность выделить такой класс.

3.5. **Теоремы восстановления.** Рассмотрим адресное обращение (2.1). Предположим, что, воспользовавшись техникой пункта 3.3, нам удалось выделить стандартные линейные формы L_0 и R_0, такие, что $F(L_0, R_0) = g$ (вообще говоря, L_0 известно с точностью до знака; предположим, что знак фиксирован). Исходное индексное выражение L отличается от L_0 на некоторую неизвестную константу. Обозначим $L_k = L_0 + k$. Линейная форма R_k определяется из условия $F(L_k, R_k) = g$. Поэтому все возможные пары (L_k, R_k) задаются равенствами $L_k = L_0 + k$, $R_k = R_0 - kL_0 - k(k-1)/2$.

Лемма 3. *Для фиксированного $\mathbf{z} \in \Omega$ стандартные линейные формы $L_k(\mathbf{x}, \mathbf{z})$ и $R_k(\mathbf{x}, \mathbf{z})$ являются корректными индексными выражениями в треугольном массиве тогда и только тогда, когда*

$$\forall \mathbf{x} \in M(\mathbf{A}, \mathbf{b}(\mathbf{z})) : 1 \le R_k(\mathbf{x}, \mathbf{z}) \le L_k(\mathbf{x}, \mathbf{z}). \tag{3.3}$$

Для стандартных линейных форм $L_0(\mathbf{x}, \mathbf{z})$ и $R_0(\mathbf{x}, \mathbf{z})$ справедливо представление $L_0(\mathbf{x}, \mathbf{z}) = (\mathbf{l}_x, \mathbf{x}) + (\mathbf{l}_z, \mathbf{z}) + l_0$, $R_0(\mathbf{x}, \mathbf{z}) = (\mathbf{r}_x, \mathbf{x}) + (\mathbf{r}_z, \mathbf{z}) + r_0$, где $\mathbf{l}_x, \mathbf{r}_x \in \mathbf{Z}^n$; $\mathbf{l}_z, \mathbf{r}_z \in \mathbf{Z}^p$; $l_0, r_0 \in \mathbf{Z}$. Пользуясь теоремами пункта 2.3, обнаруживаем, что можно найти параметрический

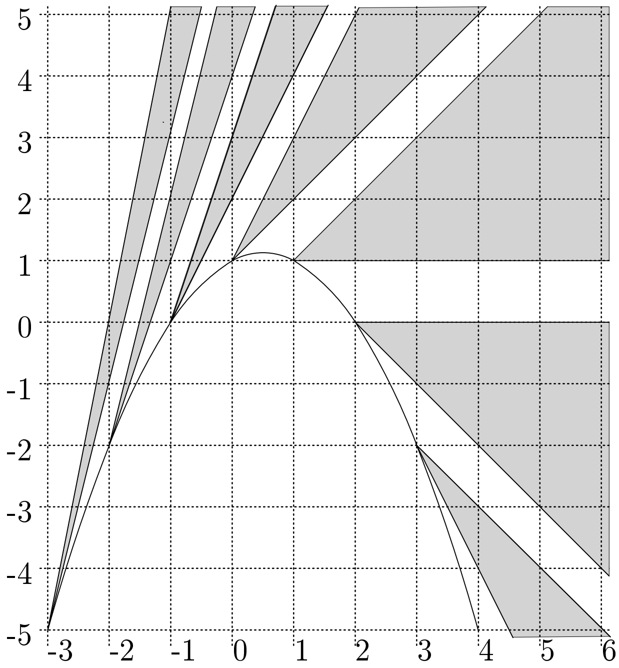

Рис. 3

минимум функций $L_0(\mathbf{x}, \mathbf{z})$ и $R_0(\mathbf{x}, \mathbf{z})$ на полиэдре $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$. Условимся обозначать минимум стандартной линейной формы $f(\mathbf{x}, \mathbf{z})$ на многограннике $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ с помощью верхнего индекса:

$$f^{min}(\mathbf{z}) = \min_{\mathbf{x} \in \{\mathbf{Ax} \leq \mathbf{b}(\mathbf{z})\}} f(\mathbf{x}, \mathbf{z}).$$

Предположим, что $f^{min}(\mathbf{z})$ всегда имеет стандартную линейную форму при $\mathbf{z} \in \Omega$. Это условие не ограничивает общности, так как в противном случае согласно (2.4) и лемме 1 можно разделить Ω на части, на каждой из которых $f^{min}(\mathbf{z})$ будет линейной формой.

Лемма 4. *Для фиксированного* $\mathbf{z} \in \Omega$ *условие* (3.3) *выполнено тогда и только тогда, когда*

$$1 \leq R_k^{min}(\mathbf{z}) \text{ и } (L_k - R_k)^{min}(\mathbf{z}) \geq 0. \tag{3.4}$$

Зададим плоскость LR с осью ординат L и осью абсцисс R. Предположим, что $\forall \mathbf{z} \in \Omega$ координаты v вершин $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ являются линейными формами от $\mathbf{z}$: $\mathbf{x}^i(\mathbf{z}) = (B^i(\mathbf{z}) + b_0^i)$, где $B^i : \mathbf{R}^p \to \mathbf{R}^n$ линейный оператор, $b_0^i \in \mathbf{R}^n$. Из леммы 1 следует, что это условие не ограничивает общности. Сопоставим точке $\mathbf{z} \in \Omega$ точку $X^i(\mathbf{z})$ на плоскости LR с координатами

$$X^i(\mathbf{z}) = \begin{pmatrix} L_0(\mathbf{x}^i(\mathbf{z}), \mathbf{z}) \\ R_0(\mathbf{x}^i(\mathbf{z}), \mathbf{z}) \end{pmatrix}.$$

X^i отображает выпуклое множество внешних параметров Ω в выпуклое множество (возможно, вырожденное в луч, отрезок или точку) $\widetilde{\Omega}$ на плоскости LR. Определим на плоскости LR множества $M_k = \{(L,R)^T : 1 \le R - kL - k(k-1)/2 \le L + k\}$. На рисунке 3 изображены области M_k для $k = \overline{-2,4}$: угол множества M_4 находится в точке $(-3,-5)^T$, угол множества M_{-2} в точке $(3,-2)^T$. Свойства множеств M_k перечислены в следующей лемме.

Лемма 5. *Справедливы следующие утверждения.*

1) M_k ограничиваются прямыми $R_k(L) = kL + k(k-1)/2$ и $R_{k+1}(L) = (k+1)L + k(k+1)/2$: $M_k = \{R_k(L) \ge 1\} \cap \{R_{k+1}(L) \le 0\}$.

2) Каждое M_k ограничивается углом, вершина которого лежит на параболе $P(L) = 1 - L(L-1)/2$ и имеет координаты $(1-k, 1-k(k-1)/2)^T$.

3) Для любых $j \ne k$ множества M_j и M_k не пересекаются.

4) Рассмотрим ограниченное множество M на плоскости LR. Множество M пересекается лишь с конечным числом M_k или не пересекается ни с одним из M_k.

5) Зададим в плоскости LR множество U с помощью прямых $U_1(L) = U_{11}L + U_{12}$ и $U_2(L) = U_{21}L + U_{22}$, $U_{11} > U_{21}$: $U = \{(L,R)^T : R \le U_{11}L + U_{12}, R \ge U_{21}L + U_{22}\}$. Множество U пересекается лишь с конечным числом M_k.

Поскольку $L_k - R_k = -R_{k+1}$, условие (3.4) можно записать в виде: $1 \le R_k^{min}(\mathbf{z})$ и $(-R_{k+1})^{min}(\mathbf{z}) \ge 0$. Введем обозначение $\Omega_k = \Omega \cap \{\mathbf{z} : 1 \le R_k^{min}(\mathbf{z})\} \cap \{\mathbf{z} : (-R_{k+1})^{min}(\mathbf{z}) \ge 0\}$.

Лемма 6. *Точка $\mathbf{z} \in \Omega_k$ тогда и только тогда, когда для всех вершин $i = \overline{1,v}$ полиэдра $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ выполнено $X^i(\mathbf{z}) \in M_k$.*

Доказательство. Подставим в неравенства, задающие M_k, координаты $X^i(\mathbf{z})$ $(i = \overline{1,v})$: $1 \le R_0(\mathbf{x}^i(\mathbf{z}), \mathbf{z}) - kL_0(\mathbf{x}^i(\mathbf{z}), \mathbf{z}) - k(k-1)/2 \le L_0(\mathbf{x}^i(\mathbf{z}), \mathbf{z}) + k$. Утверждение леммы следует из леммы 3 и известного факта из теории линейного программирования: экстремальное значение линейного функционала на полиэдре $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ достигается в некоторой вершине. $\square$

Лемма 7. *Для любых целых чисел j и k таких, что $j \ne k$, $\Omega_j \bigcap \Omega_k = \emptyset$.*

Доказательство. Предположим, что $\exists \mathbf{z} \in \Omega_j : \mathbf{z} \in \Omega_k$. Тогда для всех вершин $i = \overline{1,v}$ полиэдра $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ выполнено $X^i(\mathbf{z}) \in M_j$ и $X^i(\mathbf{z}) \in M_k$. Но множества M_j и M_k не пересекаются ввиду утверждения 3) леммы 5. $\square$

Пользуясь алгоритмами параметрического линейного программирования пункта 2.3, можно разбить Ω на непересекающиеся многогранники

Ω_k так, что $\mathbf{z} \in \Omega_k$ тогда и только тогда, когда для z и k выполнено (3.3). Для $\mathbf{z} \in \Omega_k$ искомыми индексными выражениями являются L_k и R_k. Адресное обращение (2.1) имеет конечное число вариантов восстановления линейных индексных выражений тогда и только тогда, когда лишь конечное число множеств Ω_k целочисленно не пусты. Достаточным признаком целочисленной непустоты Ω_k является непустота Ω_k. Следующие теоремы дают достаточные признаки непустоты лишь конечного числа Ω_k. Все эти теоремы имеют силу для адресных обращений с матрицей $\mathbf{A}$, которая не является абсолютно унимодулярной. Унимодулярность $\mathbf{A}$ требуется только при реализации алгоритма, описанного в пункте 2.3.

Теорема 3. *Если множество Ω внешних параметров ограничено, то не более чем конечное число множеств Ω_k непусто.*

Доказательство. Для произвольной вершины многогранника $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ множество $\widetilde{\Omega} = X^i(\Omega)$ является ограниченным. Утверждение теоремы следует из леммы 6 и пункта 4) леммы 5. □

Определение 6. Назовем *лучом* $\mathcal{L}_{\mathbf{z}_0,\mathbf{v}}$ множество $\mathcal{L}_{\mathbf{z}_0,\mathbf{v}} = \{\mathbf{z} \in \mathbf{R}^p : \mathbf{z} = \mathbf{z}_0 + \alpha\mathbf{v}, \alpha \geq 0\}$. Будем называть точку $\mathbf{z}_0$ *основанием* луча, а вектор $\mathbf{v}$ — *направлением* луча. Из каждого основания луч может выходить по нескольким направлениям. Будем обозначать через $V(\mathbf{z}_0)$ множество направлений всех лучей, которые выходят из основания $\mathbf{z}_0$.

Лемма 8. *Произвольный неограниченный полиэдр можно представить в виде объединения лучей так, что множество Ω^z оснований лучей ограничено, а каждое направление луча из множества $V(\mathbf{z}_0)$ имеет единичную норму:*

$$\Omega = \bigcup_{\mathbf{z}_0, v} \mathcal{L}_{\mathbf{z}_0,\mathbf{v}},\ \mathbf{z}_0 \in \Omega^z\text{: } ||\mathbf{z}_0|| < \beta, \quad \mathbf{v} \in V(\mathbf{z}_0),\ {\scriptstyle ||\mathbf{v}||=1.} \tag{3.5}$$

Эта лемма следует из [7, Глава 2, 1.4, Следствие 3].

Теорема 4. *Пусть существует такая вершина i полиэдра $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$, что для всех $\mathcal{L}_{\mathbf{z}_0,\mathbf{v}}$ в объединении* (3.5)

1) $(\mathbf{l}_x, B^i(\mathbf{v})) + (\mathbf{l}_z, \mathbf{v}) \geq \varepsilon > 0$,

2) $L_0(\mathbf{x}^i(\mathbf{z}_0), \mathbf{z}_0) \geq 1$.

Тогда не более чем конечное число множеств Ω_k непусто.

Замечание 1. Условия теоремы требуют, чтобы все лучи $(\mathbf{x}^i(\mathcal{L}_{\mathbf{z}_0,\mathbf{v}}), \mathcal{L}_{\mathbf{z}_0,\mathbf{v}})$ принадлежали *квадранту* (части пространства, ограниченной двумя гиперплоскостями, не параллельными $L_0(x, z) = 1$), который лежит в положительном полупространстве относительно плоскости $L_0(x, z) = 1$.

Доказательство. В условиях теоремы лучу $(\mathbf{x}^i(\mathcal{L}_{\mathbf{z}_0,\mathbf{v}}), \mathcal{L}_{\mathbf{z}_0,\mathbf{v}})$ сопоставляется луч $\mathcal{L}_{\tilde{\mathbf{z}}_0,\tilde{\mathbf{v}}}$ по правилу

$$\tilde{\mathbf{z}}_0 = X(\mathbf{z}_0),\ \tilde{\mathbf{v}} = \begin{pmatrix} \tilde{v}_L \\ \tilde{v}_R \end{pmatrix} = \begin{pmatrix} (\mathbf{l}_x, B^i(\mathbf{v})) + (\mathbf{l}_z, \mathbf{v}) \\ (\mathbf{r}_x, B^i(\mathbf{v})) + (\mathbf{r}_z, \mathbf{v}) \end{pmatrix}. \tag{3.6}$$

Из условия теоремы следует, что

$$\tilde{v}_L = (\mathbf{l}_x, B^i(\mathbf{v})) + (\mathbf{l}_z, \mathbf{v}) \geq \varepsilon > 0. \tag{3.7}$$

Из представления (3.5) получаем следующее представление для $\widetilde{\Omega} = X^i(\Omega)$ ($\widetilde{\Omega} \subset LR$):

$$\widetilde{\Omega} = \bigcup_{\tilde{\mathbf{z}}_0,\tilde{\mathbf{v}}} \mathcal{L}_{\tilde{\mathbf{z}}_0,\tilde{\mathbf{v}}},\ {}^{\tilde{\mathbf{z}}_0 \in X^i(\Omega^z):||\tilde{\mathbf{z}}_0||<\tilde{\beta}, \tilde{\mathbf{v}} \in \tilde{V}(\tilde{\mathbf{z}}_0)}_{0<\gamma\leq\sqrt{(\tilde{v}_L)^2+(\tilde{v}_R)^2}\leq\delta}. \tag{3.8}$$

Ввиду (3.7) множество $\widetilde{\Omega}$ включено в некоторый угол, который задается прямыми $U_1(L) = (\delta/\varepsilon)L + U_{12} \leq 0$ и $U_2(L) = -(\delta/\varepsilon)L + U_{22} \geq 0$. Утверждение теоремы следует из пункта 5) леммы 5 и леммы 6. ☐

Теорема 5. *Пусть существует такая вершина i, что для всех лучей $\mathcal{L}_{\mathbf{z}_0,\mathbf{v}}$ в объединении* (3.5)

1) $(\mathbf{l}_x, B^i(\mathbf{v})) + (\mathbf{l}_z, \mathbf{v}) = 0$,

2) $(\mathbf{r}_x, B^i(\mathbf{v})) + (\mathbf{r}_z, \mathbf{v}) = 0$.

Тогда не более чем конечное число множеств Ω_k непусты.

Доказательство. Из условий теоремы вытекает, что множество $\widetilde{\Omega}$ является ограниченным на плоскости LR: ввиду (3.6), $\widetilde{\Omega} = X^i(\Omega) = X^i(\Omega^z)$. Утверждение теоремы следует из пункта 4) леммы 5 и леммы 6. ☐

Теорема 6. *Пусть выполнены условия 1) и 2) предыдущей теоремы. Если в дополнение к ним выполнено условие*

3) для любого $\mathbf{z}_0 \in \Omega^z$: $L_0(\mathbf{x}^i(\mathbf{z}_0), \mathbf{z}_0) = U_1$, $R_0(\mathbf{x}^i(\mathbf{z}_0), \mathbf{z}_0) = U_2$, где U_1 и U_2 — некоторые константы,

то не более чем одна из областей $\{\Omega_k\}$ непуста.

Доказательство. Из условий теоремы следует, что вершина i отображается в точку $(U_1, U_2)^T$ плоскости LR. Поскольку множества M_k не пересекаются, то точка $(U_1, U_2)^T$ может попасть не более чем в одно из них. Воспользовавшись леммой 6, получаем справедливость утверждения теоремы. ☐

Теорема 7 (принцип неподвижной точки пространства итераций)**.** Пусть выполнены следующие условия.

1) Хотя бы одна вершина $M(\mathbf{A}, \mathbf{b}(\mathbf{z}))$ имеет фиксированные координаты, не зависящие от внешних параметров.

2) $\mathbf{l}_z = 0$ и $\mathbf{r}_z = 0$; это означает, что L_0 и R_0 зависят только от параметров объемлющих циклов и не зависят от внешних переменных.

Тогда не более чем одна из областей $\{\Omega_k\}_{k=-\infty}^{\infty}$ непуста.

Доказательство. Проверим выполнение условий теоремы 6. Обозначим координаты вершины через $\mathbf{x}^i$. Поскольку вершина имеет координаты, не зависящие от внешних параметров, то $B^i = 0$. Следовательно, для всех $\mathcal{L}_{\mathbf{z}_0,\mathbf{v}}$ в объединении (3.5) $(\mathbf{l}_x, B^i(\mathbf{v})) + (\mathbf{l}_z, \mathbf{v}) = (\mathbf{l}_x, 0) + (0, \mathbf{v}) = 0$, и $(\mathbf{r}_x, B^i(\mathbf{v})) + (\mathbf{r}_z, \mathbf{v}) = (\mathbf{r}_x, 0) + (0, \mathbf{v}) = 0$. Поскольку L_0 и R_0 зависят только от параметров цикла, то для любого $\mathbf{z}_0 \in \Omega^z$: $L_0(\mathbf{x}^i(\mathbf{z}_0), \mathbf{z}_0) = (\mathbf{l}_x, \mathbf{x}^i) + (\mathbf{l}_z, \mathbf{z}_0) + l_0 = (\mathbf{l}_x, \mathbf{x}^i) + l_0 = U_1$ и $R_0(\mathbf{x}^i(\mathbf{z}_0), \mathbf{z}_0) = (\mathbf{r}_x, \mathbf{x}^i) + (\mathbf{r}_z, \mathbf{z}_0) + r_0 = (\mathbf{r}_x, \mathbf{x}^i) + r_0 = U_2$. Следовательно, условия теоремы 6 выполнены и теорема 7 справедлива. $\square$

Замечательная особенность последней теоремы состоит в том, что ее условия можно легко проверить исходя сразу из текста программы, не обращаясь к довольно сложному построению леммы 1. Есть основания полагать, что условия теоремы 7 выполнены для подавляющего большинства реальных программ класса VD, обрабатывающих упакованный треугольный массив.

3.6. **Согласование адресных обращений.** После исследования всех адресных обращений необходимо их согласование. Мы затронем только две проблемы для иллюстрации возможного пути решения остальных.

1) *Нахождение размера используемой части массива.* Мы можем решать эту задачу, используя технику пункта 2.3 для нахождения максимума $L_k(\mathbf{x}, \mathbf{z})$.

2) *Согласование с линейными формами.* Пусть мы восстановили какое-то адресное обращение к массиву P с нелинейной $g(\mathbf{x}, \mathbf{z}) = F(L(\mathbf{x}, \mathbf{z}), R(\mathbf{x}, \mathbf{z}))$. Может возникнуть такая ситуация, что в другом адресном обращении к массиву P функция $g_1(\mathbf{x}, \mathbf{z})$ имеет стандартную линейную форму. Логично предположить, что g_1 имеет представление $g_1 = F(a, R_1)$, где $a \geq 1$. Это один из случаев, упоминавшихся в классификации пункта 3.2. В этой ситуации справедливы все теоремы пункта 3.5 с $L_0 = 1$, $R_0 = g_1$.

4. Пример

Было бы неинтересно применять идеи статьи к тривиальной подпрограмме S2 на рисунке 1. Поэтому разберем преобразование адресных обращений у реальной небольшой программы, хотя все наши результаты применимы к большим сложным программам со сложными циклическими конструкциями. Рассмотрим подпрограмму DPPFA из пакета LINPACK. Подпрограмма

DPPFA методом квадратного корня (Холецкого) находит LU-разложение самосопряженной положительно определенной матрицы, причем матрица хранится в упакованном треугольном массиве. После подстановки индексных функций вместо индуктивных переменных (см. введение) мы получаем 7 нелинейных адресных обращений к массиву `AP`. Исследуем их в порядке следования по программе.

1. $F = J(J-1)/2 + K$, $K \in \overline{1, J-1}$, $J \in \overline{2, N}$, $\Omega = \{\overline{2, \infty}\}$. $L_0 = J$, $R_0 = K$. У многогранника ограничений есть фиксированная вершина $(1, 2)$, индексы L_0 и R_0 не зависят от внешних параметров. Из теоремы 5 следует, что можно предъявить не более одной непустой области Ω_k. $K^{min} = 1$, $(J-K)^{min} = 1$ (при $J = 2$, $K = 1$), $\Omega_0 = \Omega = \{\overline{2, \infty}\}$, следовательно, исходные индексы $L = J$ и $R = K$.

2. $F = K(K-1)/2+1$, $K \in \overline{1, J-1}$, $J \in \overline{2, N}$, $\Omega = \{\overline{2, \infty}\}$. $L_0 = K$, $R_0 = 1$. $\Omega_0 = \Omega = \{\overline{2, \infty}\}$, исходные индексы: $L = K$ и $R = 1$.

3. $F = J(J-1)/2 + 1$. Аналогично предыдущему: $L = J$, $R = 1$, $\Omega_0 = \Omega$.

4. $F = K(K-1)/2 + K$, $K \in \overline{1, J-1}$, $J \in \overline{2, N}$, $\Omega = \{\overline{2, \infty}\}$. $L_0 = K$, $R_0 = K$. Результат: $L = K$, $R = K$, $\Omega_0 = \Omega$.

5. $F = J(J-1)/2 + K$. Все дословно совпадает с шагом 1.

6. $F = J(J-1)/2 + J$. $J \in \overline{1, N}$, $\Omega = \{\overline{1, \infty}\}$. $L_0 = J$, $R_0 = J$. $\Omega_0 = \Omega = \{\overline{1, \infty}\}$, исходные индексы: $L = J$ и $R = J$.

7. $F = J(J-1)/2 + J$. Совпадает с шагом 6.

Теперь можно воспользоваться методом, предложенным во введении, для того, чтобы свести фрагмент к линейному классу.

5. Заключение

Была представлена техника, которая может быть использована для расширения традиционных рамок статического анализа программ. В несколько видоизмененном виде ее можно применять к программам, неявно работающим со стандартными многомерными массивами, что позволит применить статический анализ к программам на языке Си, которые используют изменяющиеся указатели.

Автор признателен Вл.В. Воеводину за постановку задачи, критику и внимание к работе. Совместные дискуссии с В.В. Воеводиным оказались весьма плодотворными и помогли придать результатам законченную форму. Несколько существенных замечаний сделал А.В. Фролов. Автор благодарен П.В. Харитонову за то, что он внимательно прочитал окончательный вариант и предложил ряд поправок.

Список литературы

[1] *Воеводин В.В.* Математические основы параллельных вычислений. М.: Изд-во МГУ, 1991, 345 С.

[2] *Voevodin V.V.* Mathematical foundations of parallel computing, Singapur: World Scient. Publ. Co., Ser. Computer Sc. 1992, Vol. 33, 343 P.

[3] *Voevodin V.V.* Information structure of sequental programs // Russ. J. of Num. An. and Math. Modelling, 1995, Vol.10, $N^0 3$, P.279–286.

[4] *Воеводин Вл.В.* Теория и практика исследования параллелизма последовательных программ//Программирование. 1992. $N^0 3$. С. 38–53.

[5] *Pugh W.* A Practical Algorithm for Exact Array Dependence Analysis// Communs ACM. 1992. V.35.$N^0 8$. P.102–114.

[6] *Ашманов С.А., Тимохов А.В.* Теория оптимизации в задачах и упражнениях. М.: Наука. Физматгиз, 1991.

[7] Мину М. Математическое программирование. М.: Наука, Физматгиз, 1990, 488 С.
Minoux M. Paris: Programmation Matématique. Bordas et C.N.E.T—E.N.S.T., 1989.

[8] *Ковалев М.М.* Дискретная оптимизация. Минск: Изд-во БГУ, 1977.

[9] *Гофман А.Дж., Краскал Дж.Б.* Целочисленные граничные точки выпуклых многогранников.//Линейные неравенства и смежные вопросы. М.: Изд-во иностр. лит., 1959, С. 325–347.

[10] *Корбут А. А., Финкельштейн Ю.Ю.* Дискретное программирование. М.: Наука, 1969.

[11] *Хеллер И., Томпкинс Ч.Б.* Обобщение одной теоремы Данцига.//Линейные неравенства и смежные вопросы. М.: Изд-во иностр. лит., 1959, С. 348–351.

[12] *Кнут Д.* Искусство программирования для ЭВМ. Т.1. Основные алгоритмы, М.: Мир, 1976.

- Перевод названия на английский язык:
  “Applying linear index expressions reconstruction for reducing programs to the linear class.”
- Подпись автора: .... /Д.В.Хмелёв/
- Дата отправки статьи: 10 июня 1997 года.
- Должность: студент 4 курса м/м МГУ им. М.В.Ломоносова, кафедра теории вероятностей.
- Домашний адрес: 117234, Воробьевы горы, ГЗ МГУ, Б-841.
- E-mail: dima@vvv.srcc.msu.su.