Р. А.Осмонов, Д. Н. Штокало

ПРЕОБРАЗОВАНИЯ ЦИКЛОВ, ОСНОВАННЫЕ НА НЕСИНГУЛЯРНЫХ МАТРИЦАХ

ВВЕДЕНИЕ

В статье предложен способ использования несингулярных матриц вместо унимодулярных для преобразований гнезд циклов. Несингулярные матрицы, детерминант которых не ограничен значением ±1, включают унимодулярные матрицы как частный случай и дают возможность включить новое преобразование, называемое в системе несингулярного преобразования масштабированием цикла. Полученный результат показывает, что проще работать с несингулярными матрицами, чем с унимодулярными.

Главное преимущество унимодулярного преобразования состоит в том, что оно дает подход к решению так называемой «упорядоченности преобразований» для многих задач, в которых не существует явного порядка выполнения преобразований. Можно получить унимодулярную матрицу, из которой может быть просто определен нужный порядок преобразований цикла [1, 2].

Проще сгенерировать несингулярную матрицу, чем унимодулярную, так как во втором случае существует несколько ограничений, которые должны быть удовлетворены. В этой статье приводится процедура, которая генерирует несингулярную матрицу, данную изначально с первыми несколькими строками. Эта процедура нетривиальная, так как необходимо обеспечить, чтобы результирующая матрица сохраняла зависимости в гнезде цикла. В общем случае генерация преобразованного гнезда цикла более проста при использовании несингулярных матриц, чем при использовании унимодулярных, и это главная идея статьи.

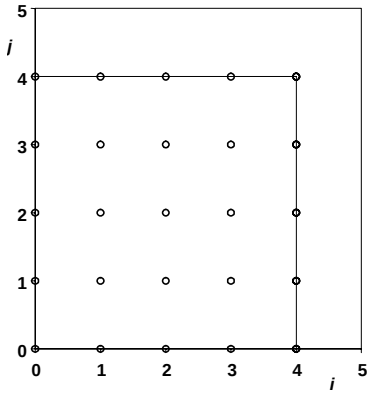
1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Гнездо цикла на рис. 1(б) имеет следующее свойство: каждая целочисленная точка в пределах границ цикла — это точка в пространстве итераций гнезда цикла. Такое пространство называется *плотным* пространством итераций. Для сравнения, в *разреженном* пространстве итераций целочис-

ленная точка может находиться в пределах границ цикла, но не быть итерационной точкой в пространстве итераций цикла. Понятие плотности и разреженности может быть формально определено следующим образом [3].

Определение 1. Пространство итераций плотное, если для любых двух целочисленных векторов v_1 и v_2, представляющих итерации цикла, любой целочисленный вектор $v_3 = \lambda v_1 + (1 - \lambda)v_2$ для некоторого $0 \le \lambda \le 1$ также представляет итерацию цикла. Пространство итераций разреженное, если оно не плотное.

```
for (i1=0; i1<5; i1++)
      for (i2=0; i2<5; i2++)
      {
          a[i1+1][i2]=b[i1][i2]+c[i1][i2];
          b[i1][i2+1]=a[i1][i2+1]+1;
      }
```



(*а*) (*б*)

Рис. 1. Гнездо цикла (а), плотное пространство итераций (б)

Смысл этой классификации пространства итераций состоит в том, что значительно труднее сгенерировать код для гнезда цикла в случае, когда пространство итераций разрежено, чем если пространство итераций является плотным, как будет показано в разд. 3.

1.1. Преобразования цикла

В этой статье рассматриваются преобразования, которые могут быть представлены линейными идентичными отображениями из пространства итераций исходной программы в пространство итераций конечной программы. Эта система преобразований включает перестановку, скашивание и обращение, а также новое преобразование — масштабирование цикла. Все эти преобразования являются стандартными за исключением масштабирования. Масштабирование цикла — преобразование, приводящее к разреживанию пространства итераций, и как следствие, — к изменению шага

цикла. Изменению подвергается внутренний цикл. Масштабирование цикла изменяет размер шага цикла, масштабируемая матрица не является унимодулярной. Линейное идентичное отображение между итерационными пространствами может быть получено при использовании целочисленных несингулярных матриц. Обратно, можно показать, что преобразование, представленное любой целочисленной несингулярной матрицей, может быть показано как композиция четырех основных преобразований. Более строго это может быть выражено в виде следующей теоремы.

Теорема 1. Преобразование, представленное любой целочисленной несингулярной матрицей, может быть представлено как композиция перестановки, скашивания, обращения и масштабирования.

Доказательство. Путем применения соответствующих элементарных строковых и столбцовых операций, целочисленная несингулярная матрица может быть приведена к диагональной матрице [4]. Элементарные операции можно представить путем умножения перестановочной, обращающей и скашивающей матриц с правой и с левой сторон несингулярной матрицы.

Теорема 2. Унимодулярные преобразования отображают плотное (разреженное) пространство итераций в другое плотное (разреженное) пространство итераций.

Доказательство. Пространство остается таким же, меняется только базис [5].

1.2. Генерация кода

Чтобы сгенерировать код для конечного гнезда цикла, необходимо сгенерировать циклы, которые просматривают точки конечного пространства итераций в лексикографическом порядке и переводят в операторах тела цикла старые индексы цикла к новым индексам цикла.

Если вектора S_i и S_j представляют начальные и конечные итерационные переменные, тогда имеем $S_i = T^1 S_j$. Это множество уравнений выражает старые индексы в терминах новых. Они могут быть использованы для замены исходных индексных переменных в теле цикла на новые индексные переменные. Для текущего примера это множество уравнений следующее:

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} \frac{1}{5} & -\frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}.$$

Заметим, что $T^{-1}S_j$ всегда будет целочисленной точкой, даже в случае, если элементы T^{-1} являются рациональными.

2. ЗАДАЧИ ГЕНЕРАЦИИ КОДА

Проблема при генерации циклов для получения конечного пространства итераций заключается в том, что данное преобразование, в основном, не сохраняет лексикографический порядок (две итерации могут быть выполнены в одном порядке в исходном гнезде цикла, но в другом порядке — в конечном гнезде цикла). Поэтому не существует явного пути для использования исходного гнезда цикла, чтобы сгенерировать код. В первую очередь, можно найти отображение исходных границ и затем сгенерировать гнездо цикла, которое посещает в лексикографическом порядке все целочисленные точки в области, ограниченной отображением. Этот подход работает хорошо, когда конечное пространство итераций плотное, разреженное пространство итераций потребует дополнительного аппарата, который позволил бы перепрыгивать через точки, не входящие в пространство итераций.

2.1. Вычисление отображения границ

Существует много путей для подсчета отображения исходных границ, один из простых методов использует исключение Фурье—Моцкина.

Для вычисления соответствующих границ цикла в данной работе используется алгоритм исключения Фурье—Моцкина [5, 6]. Алгоритм метода исключения Фурье—Моцкина достаточно прост. Рассмотрим систему линейных неравенств

$$\sum_{j=1}^{n} a_{ij} x_j \le b_j, i = (1,...,m).$$

Эта система может быть представлена в виде неравенств в соответствии со знаком коэффициента x_n.

$$x_n \le D_i(\overline{x}), i = (1,...,p),$$
$$x_n \ge E_j(\overline{x}), j = (1,...,q),$$
$$0 \le F_k(\overline{x}), k = (1,...,r),$$

где D_i, E_j и F_k — линейные функции от $\overline{x} = (x_1,..x_{(n-1)})$.

Теперь можно исключить x_n из системы для получения следующей преобразованной системы:

$$E_j(\bar{x}) \le D_i(\bar{x}), i = (1,...,p), j = (1,...,q),$$
$$0 \le F_k(\bar{x}), k = (1,...,r).$$

Этот процесс повторяется до тех пор, пока останется только одна переменная. Границы для этой переменной могут быть определены путем рассмотрения преобразованной системы уравнений.

Возвращаясь к задаче, рассмотрим систему неравенств для S_j. Границы цикла для j_n могут быть вычислены посредством решения неравенств относительно j_n. Границы для j_k могут быть вычислены путем исключения $j_{(k+1)},...,j_n$ из системы, используя метод исключения Фурье—Моцкина и т.д.

Рассмотрим данный пример. Пространство итераций на рис. 1(б) ограничено системой линейных неравенств:

$$0 \le i1 \le 4,$$
$$0 \le i2 \le 4.$$

Вычислим *i, j* в терминах *u, v*, используя матрицу преобразования, затем, заменяя *i, j* на *u, v* в неравенствах и применяя метод исключения Фурье—Моцкина, получим следующую картину границ пространства итераций:

$$0 \le u \le 4,$$
$$max(-2u, (u-20)/2) \le v \le min(20 - 2u, u/2).$$

К сожалению, нельзя использовать эти неравенства непосредственно при генерации кода для конечного гнезда цикла. Существуют две проблемы. Первая заключается в том, что нижняя и верхняя границы могут быть даже нецелочисленными, например, когда $u = 3$, верхняя граница для $v = 3/2$. Вторая заключается в том, что, даже если бы начальное пространство итераций было плотным, конечное пространство итераций будет разреженным. Это означает, что необходимо найти некоторые пути для перепрыгивания точек, которые не представляют итерации в пространстве конечного гнезда цикла.

2.2. Плотное пространство итераций

Для частного случая, когда конечное пространство итераций является плотным (в случае использования унимодулярной матрицы для преобразования гнезда цикла с плотным пространством итераций (теорема 2)), обе эти задачи могут быть решены просто. Если конечное пространство итераций плотное, нет необходимости перепрыгивать через точки, которые не

входят в пространство итераций конечного гнезда цикла. Более того, можно использовать операции «нижняя целая часть» и «верхняя целая часть» для получения ближайших целых чисел в пределах отображения границ.

2.3. Разреженное пространство итераций

Для разреженного пространства итераций операции «нижняя целая часть» и «верхняя целая часть» не могут решить проблему. Например, на рис. 6 (б) *(5, –7)* — это целочисленная точка, ближайшая к границе v при $u = 5$, но начальная точка, соответствующая $u = 5$, есть $v = 5$. В этом случае возможно только использование условного теста в теле цикла для избежания выполнения тела цикла в точках, которые не соответствуют точкам конечного пространства итераций. Условный тест предполагает посещение целочисленных точек, в которых нет необходимости, кроме того, условный тест дорог.

3. АЛГОРИТМ ГЕНЕРАЦИИ КОДА

Основная идея в понимании решения главной задачи состоит в том, что целочисленная несингулярная матрица T может быть разложена на составные части в виде произведения нижней треугольной матрицы H с положительными диагональными элементами и унимодулярной матрицы U. Это разложение относится к нормальным формам Эрмита матричного преобразования [7]. Действительно, если U используется для преобразования программы, результирующая программа выполняет итерации в том же самом лексикографическом порядке, как и программа, полученная путем использования T как матрицы преобразования. Можно заметить, что диагональные элементы H соответствуют размерам шагов цикла. Данное разложение дает алгоритм, который генерирует эффективный код для общего случая несингулярных матриц.

3.1. Вычисление матрицы полного преобразования

Данная процедура дополняет матрицу частичного преобразования до несингулярной квадратной, используя матрицу векторов зависимости, поэтому требуется совпадение количества строк матрицы частичного преобразования с её рангом.

В первом цикле удаляются некоторые столбцы матрицы зависимости, далее не участвующие в алгоритме. Во втором цикле процедуры происхо-

дит поиск некоторого вектора, линейно независимого от всех строк существующей на данный момент матрицы частичного преобразования, отклоненного от векторов матрицы зависимости не более чем на 90 градусов. Этот вектор дополняет матрицу частичного преобразования, причем соответствующие ему по определённому правилу вектора матрицы зависимости вычеркиваются из нее. Данная техника применяется до тех пор, пока матрица зависимости не пуста. На завершающей стадии алгоритма выполняется произвольное дополнение матрицы частичного преобразования до несингулярной квадратной.

На входе: транспонированная $m \times n$ матрица частичного преобразования $P^T_{m \times n}$, матрица зависимостей $D_{n \times n}$.

На выходе: матрица полного преобразования $T_{n \times n}$.

for i=1 to m {
$f^T := P_i^T D$;
Вычёркиваем из матрицы D столбцы D^j, где $f_j > 0$;
}
r:=m;
while D не пуста {
Пусть k — первая ненулевая строка D;
$x := c(I - P(P^T P)^{-1} P^T)e_k$, где число $c > 0$, приводящее x к вектору целых чисел с взаимопростыми компонентами;
Вычёркиваем из D столбцы D^j, где $D_{kj} > 0$;
Дополняем матрицу $P^T_{r+1} = x^T$;
r=r+1;
}
Пусть $R_{n \times n}$ — матрица перехода, полученная в результате работы алгоритма *HermiteNormalForm(P^T) (рис. 3)*;
Дополняем матрицу $P^T_{r \times n}$ до квадратной присоединением вниз нижних строк матрицы $R_{n \times n}$
($P^T_{n \times n} = appendBottom(P^T_{r \times n}, R[r+1:n, 1:n])$;

Результат $T := P^T_{n \times n}$.

Рис. 2. Алгоритм генерации матрицы

Несингулярная матрица $T = \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$, сгенерированная алгоритмом в соответствии с матрицей зависимости гнезда цикла $D = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$.

3.2. Вспомогательное пространство итераций

С помощью применения операций со столбцами к целочисленной несингулярной матрице T можно привести ее к целочисленной нижней треугольной матрице с положительными диагональными элементами [8]. Эта нижняя треугольная матрица относится к нормальной форме Эрмита [4, 7] матрицы T. Из этого следует, что T может быть записана как произведение нижней треугольной матрицы H с положительными диагональными элементами и унимодулярной матрицы U, которая представляет композицию операций со столбцами. Это разложение не уникальное, но для преобразования цикла можно использовать любое такое разложение.

Рис. 4 показывает алгоритм вычисления H и U [9].

Пусть $T = HU$, S_i — исходное пространство и S_j — конечное пространство. Определим $S_k = US_i$. Тогда $S_j = TS_i = HUS_i = HS_k$

Определение 2. Пространство итераций S_k называется *вспомогательным* пространством итераций к S_i по отношению к разложению HU.

Теорема 3. Вспомогательное пространство итераций является плотным пространством итераций, если начальное пространство плотное.

Доказательство: Следует из Теоремы 2, так как U — унимодулярная матрица.

Важное свойство вспомогательного пространства заключается в том, что оно выполняет итерации в том же лексикографическом порядке, что и конечное пространство итераций.

3.3. Нахождение формы Эрмита

Алгоритм представляет стандартную процедуру приведения матрицы к треугольному виду. Специфика алгоритма заключается в сохранении целочисленности элементов матрицы.

На входе: целочисленная матрица $T_{n\times n}$.
На выходе: форма Эрмита H матрицы T, матрица перехода $U_{n\times n}$ такая, что $HU=T$.

Пусть $U:=I$ — единичная матрица;
for i=1 to n {
while $T[i, i+1:n]\neq\vec{0}$ {
Найдём наименьший по модулю отличный от нуля элемент t_{ij} вектора $T[i, i : n]$;
if $i \neq j$ then меняем i,j столбцы T и i,j строки U ($T = TU_{ij}$, $U = U_{ij}^{-1} U$);
for j=i+1 to n {
Складываем столбец $[-\frac{t_{ij}}{t_{ii}}]\, T_i$ со столбцом T_j ($T = TU_{ij}$, $U = U_{ij}^{-1} U$);
}
}
}
$H:=T$;
Результат: матрицы H,U.

Рис. 3. Алгоритм вычисления формы Эрмита

Важное свойство вспомогательного пространства заключается в том, что оно выполняет итерации в том же самом лексикографическом порядке, как и конечное пространство итераций.

$$T=\begin{pmatrix}2 & 1\\ 1 & -2\end{pmatrix}\ H=\begin{pmatrix}1 & 0\\ -2 & 5\end{pmatrix}\ U=\begin{pmatrix}2 & 1\\ 1 & 0\end{pmatrix}.$$

Рассмотрим унимодулярное преобразование $U = \begin{pmatrix}2 & 1\\ 1 & 0\end{pmatrix}$ на текущем примере.

$$\begin{pmatrix}p\\ q\end{pmatrix}=U\begin{pmatrix}i\\ j\end{pmatrix}.$$

Для исходных границ на рис. 1(а) можно вычислить отображение границ, показанное на рис. 4(а). Так как вспомогательное пространство плотное, можно использовать операции «нижняя целая часть» и «верхняя целая часть» для подсчета точных границ.

Теорема 4. Если вспомогательное пространство итераций обходится в лексикографическом порядке, тогда конечное пространство итераций также обходится в лексикографическом порядке.

Доказательство. $S_j = HS_k$, где H — нижняя треугольная матрица с положительной диагональю. Пусть $\vec{k}_1 \prec \vec{k}_2$ — две итерации во вспомогательном пространстве итераций, $\vec{d}_1 = \vec{k}_2 - \vec{k}_1$ — разница этих двух векторов. Ясно, что $\vec{d}_1 \succ 0$. Чтобы увидеть, что лексикографический порядок сохраняется, рассмотрим новый дистанционный вектор $\vec{d}_2$

$$\vec{d}_2 = \vec{j}_2 - \vec{j}_1 = H\vec{k}_2 - H\vec{k}_1 = H\vec{d}_1 .$$

Если $\vec{d}_1(i)$ – i-й элемент $\vec{d}_1$, то ведущий ненулевой элемент $\vec{d}_1(i)$ должен быть положительным, так как $\vec{d}_1 \succ 0$. Тогда ведущий ненулевой элемент $\vec{d}_2$ — это $h_{ii}\vec{d}_1(i)$, который также положителен. Следовательно, $\vec{d}_2 \succ 0$ и $\vec{j}_1 \prec \vec{j}_2$.

Этот результат представляет собой способ генерации кода. Разложение T на HU — это генерация циклов для прохождения вспомогательного пространства с использованием унимодулярной матрицы и вычислением переменных конечного пространства итераций в теле цикла.

Используя границы начального пространства, получаем промежуточный код для текущего примера, показанный на рис. 4.

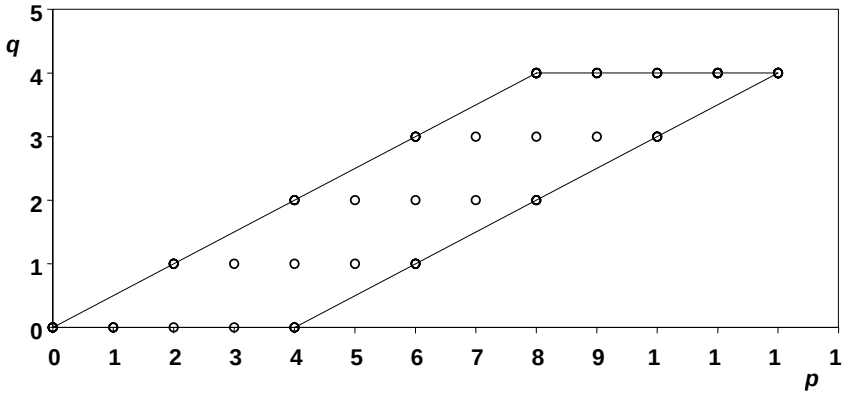
Данный код избегает выполнения условного теста, и его можно значительно улучшить. Можно заметить, что вычисление u постоянно во внутреннем цикле, кроме того, u — это линейная функция индекса внешнего цикла, и она может быть уменьшена в силе. Подобным образом, v — это линейная функция от p и q, и она может быть уменьшена в силе. Данные преобразования могут быть оставлены для дальнейшей оптимизационной стадии, предпочтительно использовать индукционные переменные u и v непосредственно как переменные цикла вместо p и q.

```
for (p=0; p<12; p++)
    for (q=max(0, ⌈ (p-4)/2 ⌉); q<min(4, ⌊ p/2 ⌋); q++)
    {
    /*функция перехода от промежуточных итерационных переменных к конечным */
```

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -2 & 5 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}$$

```
    a[(v+2*u)/5+1][(u-2*v)/5]=b[(v+2*u)/5][(u-2*v)/5]+c[(v+2*u)/5][(u-
    2*v)/5];
    b[(v+2*u)/5][(u-2*v)/5+1]=a[(v+2*u)/5][(u-2*v)/5+1]+1;
    }
```

(*а*)



(*б*)

Рис. 4. Промежуточное гнездо цикла (а),
вспомогательное пространство итераций (б)

3.4. Конечное пространство итераций

Так как H — это нижняя треугольная матрица, то легко преобразовать границы цикла вспомогательного пространства в границы цикла конечного пространства. Для текущего примера отношение между этими двумя пространствами выражается следующим уравнением:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -2 & 5 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} \quad \Rightarrow \quad \begin{cases} u = p \\ v = -2p + 5q \end{cases}.$$

Из первого уравнения следует, что границы для u — это границы для p. Следовательно, границы для u следующие: $0 \le u < 12$. Из второго уравнения следует, что границы для v — это границы для q, умноженные на *5* со смещением *(–2*u)*. Границы для u постоянные, границы для v зависят только от u. Следовательно, эти границы могут быть использованы непосредственно для построения конечного гнезда цикла. Алгоритм нахождения границ приведен на рис. 5.

3.5. Вычисление границ конечного пространства итераций

На входе: форма Эрмита H, границы вспомогательного пространства S_k.
На выходе: границы конечного пространства S_j.
for i=1 to n {
Вычислить смещение путем замены $k_1,\ldots,k_{(i-1)}$ на $j_1,..,j_{(i-1)}$
$v_i = h_{i1}k_1 + \ldots + h_{i(i-1)}k_{(i-1)}$
Вычислить нижнюю границу $l^k{}_i$, заменяя $k_1,..,k_{(i-1)}$ на $j_1,\ldots j_{(i-1)}$
$l_i^j = v_i + h_{ii}l_i^k$
Вычислить верхнюю границу $u^k{}_i$, заменяя $k_1,..,k_{(i-1)}$ на $j_1,\ldots j_{(i-1)}$
$u_i^j = v_i + h_{ii}u_i^k$
}
Результат S_j.

Рис. 5. Алгоритм нахождения границ цикла

Чтобы закончить генерацию кода, необходимо перепрыгнуть через точки в пределах границ цикла, которые не представляют итерации в конечном пространстве. Фактически оказывается, что данные точки удовлетворяют использованию циклов с постоянными размерами шагов, и эти шаги представляются числами на главной диагонали формы Эрмита.

Для текущего примера H имеет диагональ *[1, 5],* что соответствует шагу *1* для внешнего цикла и шагу *5* для внутреннего цикла. Таким образом, можно сформулировать следующую теорему.

Теорема 5. Положительные целые числа на диагонали формы Эрмита — это интервалы в каком-либо измерении.

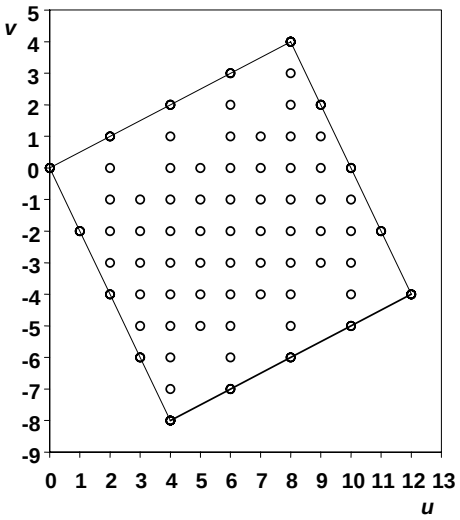
Доказательство. Рассмотрим две точки: $P1 = (k_1, k_2,\ldots k_i, k_{(i+1)}\ldots k_n)$ и $P2 = (k_1, k_2,\ldots k_i, a_{(i+1)}\ldots a_n)$ во вспомогательном пространстве, которые име-

ют одинаковые координаты в первых *(i – 1)* измерениях. Пусть *P3* и *P4* будут их отображениями в конечном пространстве. *P3* и *P4* имеют одинаковые координаты для первых *(i – 1)* измерений, кроме этого, их разница в *i*-ом измерении h_{ii}, так как H — это нижняя треугольная матрица.

Поэтому гнездо цикла может быть построено прямым прохождением конечного пространства. Для текущего примера на рис. 6 показано конечное гнездо цикла.

```
for (u=0; u<12; u++)
    for (v=-2*u+5*max⌈0,(u-4)/2⌉;  v<-2*u+5*min⌊4, u/2⌋; v=+5)
    {
        a[(v+2*u)/5+1][(u-2*v)/5]=b[(v+2*u)/5][(u-2*v)/5]+c[(v+2*u)/5][(u-2*v)/5];
        b[(v+2*u)/5][(u-2*v)/5+1]=a[(v+2*u)/5][(u-2*v)/5+1]+1;
    }
```

(а)



(б)

Рис. 6. Преобразованное гнездо цикла (а), разреженное пространство итераций (б)

Внутренний цикл в данном гнезде может быть распараллелен. Заметим, что вспомогательное пространство используется только для подсчета границ для конечного пространства.

4. СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Рассмотрим матрицу дистанционных векторов, представляющую зависимости по данным в цикле. Применим к этой матрице поочередно унимодулярное [1] и несингулярное преобразования. Полученные результаты представлены в следующей таблице.

преобразование	**D**	**T**	**H**	**U**	**D'**
несингулярное	$\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$	$\begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ -2 & 5 \end{pmatrix}$	$\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 3 & -2 \end{pmatrix}$
унимодулярное	$\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$			$\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$

Сравнивая несингулярное преобразование с унимодулярным, видим, что унимодулярное является частным случаем несингулярного. Алгоритм хорошо работает с унимодулярными преобразованиями, которые рассматриваются как частный случай. Если в качестве преобразования T сгенерирована унимодулярная матрица, то матрица H является единичной и, следовательно, шаг цикла равен единице, т.е. преобразование масштабирования цикла отсутствует. Дистанционные вектора конечного гнезда цикла различны (**D'**), но они, тем не менее, удовлетворяют условию о распараллеливании k-го цикла.

ЗАКЛЮЧЕНИЕ

Представленная система преобразования цикла основана на целочисленных несингулярных матрицах. Идея состоит в том, что сгенерированная несингулярная матрица может быть представлена в виде произведения нижней треугольной и унимодулярной матрицы. Посредством унимодулярной матрицы начальное гнездо цикла преобразуется к промежуточному

виду. Далее при помощи нижней треугольной матрицы уточняются границы, и на выходе имеем преобразованное гнездо цикла, в котором один из циклов может быть распараллелен. Также представлен простой законченный алгоритм, который получает на входе матрицу преобразования с первыми несколькими строками и генерирует полную матрицу преобразования.

СПИСОК ЛИТЕРАТУРЫ

1. Осмонов Р. А. Повышение степени параллелизма в циклах с помощью матричных преобразований // Тез. докл. конф.-конкурса работ студентов, аспирантов и молодых ученых «Технологии Microsoft в информатике и программировании». — Новосибирск, 2005. — С. 133–134.
2. Гантмахер Ф. Р. Теория матриц. — М.: Наука, 1998.
3. Schrijver A. Theory of Linear and Integer Programming. — John Wiley & Sons, 1986.
4. Ancourt C., Irigoin F. Scanning polyhedra with DO loops // Third ACM Symp. on Principles and Practice of Parallel Programming, April 1991. —P. 39–50.
5. Banerjee U. Loop transformations for restructuring compilers. The foundations. — Kluwer Academic Publishers, 1993.
6. Nemhauser G., Wolsey L. Lnteger and Combinatorial Optimization. Wiley-Interscience series in Discrete Mathematics and Optimization. — New York: John Wiley and Sons, 1988.
7. Xue J. An algorithm to automate non-unimodular transformations of loop nest // The 5th IEEE symp. on parallel distributed processing. — Dallas: IEEE computer society press, 1993.
8. Cohen H. A course in computational algebraic number theory. — Springer-Verlag, 1996.