**Р. Н. Арапбаев**

# ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ НОВОЙ СТРАТЕГИИ ТЕСТИРОВАНИЯ*

К настоящему времени разработаны многие алгоритмы для анализа зависимостей по данным [1, 2]. В [3] выработана *новая стратегия* применения тестов для выявления зависимости по данным, в которой алгоритм стратегии состоит из серии эффективных и не дорогостоящих, имеющих линейную и полиномиальную сложность тестов на зависимость. В работе учтены результаты эмпирических и теоретических исследований анализа зависимостей по данным [4, 5], а также некоторые ограничения аналогичных работ [6–9]. В настоящей работе проведено экспериментальное сравнение результатов предложенного метода с наиболее известными стратегиями тестирования анализа зависимостей по данным, такими как Эпсилон-тест [7] и алгоритм Майдана [8]. Эксперимент проведен с использованием инструмента Petit V1.2 [10], разработанного в Мэриленд-ском университете как расширенный вариант инструмента tiny [11], и с использованием системы SUIF [12], разработанной в Стенфордском университете. Для эксперимента использованы два вида данных. Первый вид — набор тестовых научных программ NASA и PERFECT Club benchmarks [13], где каждая программа включает от 500 до 18000 строк. Второй вид — набор из 16 циклов, собранный из работ, аналогичных нашей [4, 6, 8, 9, 14, 15].

Все понятия, не определяемые в этой работе, могут быть найдены в [1, 2].

## 1. СТРАТЕГИЯ ТЕСТИРОВАНИЯ

При распараллеливании циклов одной из важных проблем является выявление зависимости по данным. Тесты на зависимость должны определять, существуют ли целочисленные решения следующей системы линейных диофантовых уравнений (1), полученной при выявлении зависимостей, удовлетворяющей ограничениям границ циклов (2):

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 07-07-12050).

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + c_1 = 0\\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + c_2 = 0\\
&\qquad\qquad \ldots\ldots\ldots\\
&a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + c_m = 0
\end{aligned}
\qquad (1)
$$

и

$$L_i \le x_i \le U_i \quad \text{где } i=1,\ldots,n \qquad (2)$$

### 1.1. Алгоритм новой стратегии

В работе [3] предлагается новая стратегия применения тестов для выявления зависимости по данным, в которой алгоритм состоит из серии эффективных и недорогостоящих тестов на зависимость.

В данной стратегии, в зависимости от значений основных параметров задачи:

- размерность массивов;
- количество вложенных циклов;
- значения коэффициентов индексных переменных и значения границ циклов,

в первую очередь выделяются часто встречающие и легко разрешимые случаи. Соответственно каждому случаю применяется один быстрый и точный тест или серия эффективных тестов.

Итак, на вход нашего алгоритма подается гнездо циклов, в котором ***r*** — количество вложенных циклов, и операторы цикла обращаются к элементам ***d***-мерного массива. Кроме того, считаются постоянными и известными значения коэффициентов индексных переменных $a_{11}, a_{12}, \ldots, a_{mn}$ и значения границ циклов $L_1, L_2, \ldots, L_n, U_1, U_2, \ldots, U_n$, где $n = 2 * \boldsymbol{r}$ и $m = d$. Задача нашего алгоритма — выявить зависимости по данным между операторами в итерациях гнезда циклов, т.е. алгоритм должен возвращать ответ «да/нет» о существовании целочисленных решений $i_1, i_2 \ldots, i_n$ системы линейных диофантовых уравнений (1), удовлетворяющих ограничениям (2).

Для этого сначала выделены часто встречающие и легко разрешимые случаи задачи зависимости по данным:

1. ***r*= 1**, ***d* = 1**, т.е. внутри единственного цикла операторы обращаются к элементам одномерного массива. В этом случае уравнение зависимости (1) выглядит так: $a_1 x_1 + a_2 x_2 = a_0$ и $L \le x_1, x_2 \le U$. Для уравнения целесообразно применить самый быстрый и точный *SIV-тест* [6].

2. $\boldsymbol{r > 1}$, $\boldsymbol{d = 1}$, уравнение зависимости имеет вид: $a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = a_0$, где $L_i \le x_i \le U_i$, $i=1,\ldots,n$. Этот случай несколько усложняет решение, следовательно, применяется серия одномерных тестов на зависимость: тест Банержи [16], I-тест [17] и IR-тест [18]. Каждый следующий тест выполняется только в том случае, если был получен неточный ответ (maybe) предыдущим тестом, кроме того, после применения теста Банержи выполняется проверка коэффициентов индексных переменных для уточнения ответов теста.

3. $\boldsymbol{d = 2}$ и имеются *сцепленные индексы*. Система уравнений зависимости имеет вид:
   $a_{1,1} x_1 + a_{1,2} x_2 + \ldots + a_{1,n} x_n = a_{1,0}$
   $a_{2,1} x_1 + a_{2,2} x_2 + \ldots + a_{2,n} x_n = a_{2,0}$
   и $L_i \le x_i \le U_i$ где $i = 1, \ldots, n$.

   Этот случай доминирует в реальных последовательных программах, но применение обычных одномерных тестов на зависимость в этом случае бесполезно, так как имеются сцепленные индексные переменные. Поэтому применяется серия многомерных тестов: λ-тест [19], многомерный I-тест [20] и модифицированный λ-тест [21]. Метод запоминания результатов предыдущих тестов и использование их для последующих тестов оптимизирует данный случай.

4. В оставшихся случаях уравнение зависимости имеет вид (1) с ограничениями (2). Каждое уравнение рассматривается в отдельности, и для него последовательно применяется серия одномерных тестов: тест Банержи, I-тест и IR-тест. Этот подход дает более точный ответ, если индексные переменные *не сцеплены.* На практике доля сцепленных индексных переменных в ссылках трехмерных массивов и выше незначительна.

Учитывая все случаи, была собрана и реализована библиотека тестов на зависимость. Библиотека состоит из следующих тестов: ZIV-тест, SIV-тест, НОД-тест, Банержи-тест, I-тест, IR-тест, λ-тест, многомерный I-тест и модифицированный λ-тест. Кроме тестов на зависимость в библиотеке имеются алгоритмы для уточнения ответов теста Банержи (см. разд. 1.2.4). Все алгоритмы имеют линейную временную сложность. Из-за высокой стоимости в библиотеку не вошли точные тесты. На рис. 1 приведена общая схема новой стратегии применения тестов на зависимость.

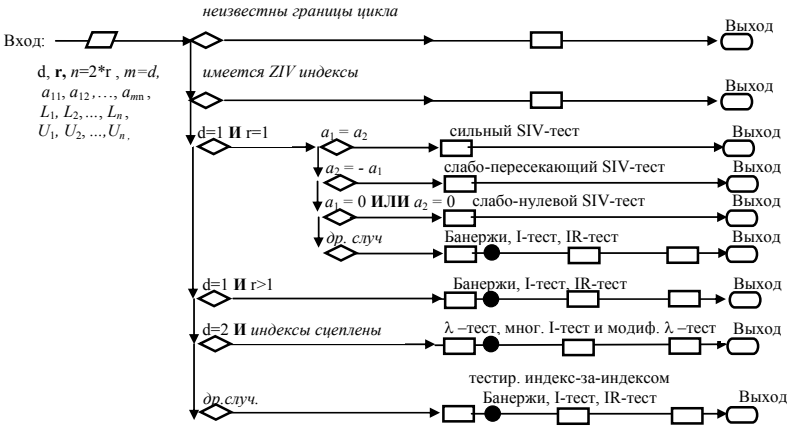



*Рис. 1.* Общая схема новой стратегии тестирования.

● — алгоритм проверки коэфц., □ — тесты на зависимость

Алгоритм новой стратегии может останавливаться в нескольких случаях, если был получен точный ответ любого теста или после применений серии одномерных тестов либо многомерных тестов на зависимость. Метод запоминания результатов предыдущих тестов и использование их для последующих тестов, а также применение алгоритмов уточняющих ответы теста Банержи, оптимизируют общее время выполнения алгоритма новой стратегии.

## 1.2. Построение новой стратегии тестирования

В этом подразделе будут показаны наиболее важные преимущества новой выработанной стратегии и некоторые требования к алгоритмам зависимостей по данным.

### *1.2.1. Аналогичные алгоритмы анализа зависимостей по данным*

Исследование алгоритмов на зависимость выявило несколько работ, в некотором отношении аналогичных к нашей работе. Во всех исследованиях был собран набор тестов на зависимость с целью использования их для ре-

шения проблемы в практических ситуациях эффективно и точно. Однако наша работа по существу отличается от них.

**Дельта-тест** [6] разработан для определенных классов ссылок массива, которые часто встречаются в научных программных кодах. Тест сначала классифицирует индексные выражения массивов на следующие категории: ZIV (нулевая индексная переменная), SIV (единственная индексная переменная) и MIV (составная индексная переменная) формы. Соответственно к каждой форме применяется быстрые и точные одноименные тесты.

В работе [7] представлен более упрощенный и быстрый вариант Дельта-теста, называемый **Эпсилон-тестом**. В этом тесте рассмотрены только самые простые случаи индексных выражений SIV, не используются *сцепленные* MIV формы и НОД-тест, а также не рассматриваются треугольные границы цикла при использовании теста Банержи. Хотя эти алгоритмы являются самыми быстрыми, но они уступают по точности предложенному в данной работе алгоритму.

**Алгоритм Майдана** [8], который использован в системе SUIF Стенфордского университета, состоит из серии точных тестов, каждый из которых применим в ограниченной области. Последний тест в алгоритме — метод исключения Фурье—Моцкина, расширенный для решения целочисленных задач. Авторы показали, что практически зависимость по данным может быть вычислена точно и эффективно. Главное различие между алгоритмом Майдана и предложенным подходом — в том, что в первом случае добивались требуемого результата с использованием дорогих методов. В противоположность этому наш подход пытается получит те же результаты с использованием более дешевых тестов на зависимость.

**K-тест** [9] также состоит из библиотеки тестов на зависимость, но в отличие от других, вместо конкретной стратегии применения тестов используются методы искусственного интеллекта, хотя в самой работе также упоминается о NP-полноте методов искусственного интеллекта.

### *1.2.2. Основные результаты существующих эмпирических исследований*

Отметим, что объектом автоматического распараллеливания служат большие научные пакеты прикладных программ, написанных на последовательных языках типа Фортран. Согласно эмпирическому изучению [4], в реальных программах индексные выражения не очень сложны. Из всех исследованных массивов примерно 56% составляют ссылки одномерных массивов и 36% — ссылки двумерных массивов. Доля ссылок трехмерных массивов и выше около 8 %. Что касается индексных выражений массивов, то 53 % являются линейными, 13 % — частично линейными и 34 % — нели-

нейными. Поэтому обычно для анализа зависимостей по данным на практике используются только одномерные тесты, использующие подход тестирования «индекс-за-индексом». В многомерных случаях система уравнений зависимости может не иметь решения даже в том случае, когда имеются решения в каждом из отдельных уравнений.

В новой стратегии для анализа ссылок одномерных массивов применяется серия из трех эффективных тестов: тест Банержи, I-тест и IR-тест.

При анализе многомерных массивов основную трудность вызывают часто встречающиеся в реальных программах *сцепленные* индексы. Как показано в эмпирических исследованиях З. Шена и др. [4] , более чем в девяти тысячах парах двумерных ссылок массивов приблизительно 46% являются сцепленными индексными выражениями. Что касается ссылок массивов большей размерности, то только 2% являются сцепленными индексными выражениями. Поэтому на практике важно иметь эффективный тест для обработки сцепленных индексов, особенно для анализа ссылок двухмерных массивов. Одним из таких эффективных алгоритмов является λ-тест.

Для обработки сцепленных индексов в ссылках двухмерных массивов, мы предлагаем применить серию многомерных тестов на зависимость: λ-тест, многомерный I-тест и модифицированный λ-тест. Что касается ссылок массива большей размерности, доля которых незначительна, то для них предлагается применять серии из одномерных тестов (тестирование “индекс-за-индексом”).

### *1.2.3. Появления новых тестов выявления зависимости по данным*

К настоящему времени разработано множество тестов на зависимости, дающие приближенные и точные решения задачи выявления зависимости по данным. Далее кратко следует описание новых тестов на зависимость, применение которых при выработке новой стратегий дало ощутимые результаты при выявлении зависимости по данным на реальных задачах.

*I–тест [17]*

Данный тест является комбинацией тестов НОД и Банержи. Как и НОД тест, он проверяет существование целочисленного решения; как и тест Банержи, он учитывает ограничения индексных переменных. I-тест преобразует каждое уравнение зависимости (1) в интервальное уравнение:

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = [L, U] \qquad (3)$$

В правой части уравнения (3) $L$, $U$ — верхняя и нижняя границы, являющиеся константами. В исходной форме верхняя и нижняя границы равны постоянному значению в правой части уравнения из системы (1). Тест в каждой итерации, выбирая переменную из левой части уравнения, перемещает ее в верхнюю и нижнюю границу в правой части (используя механизм *теста Банержи*), затем применяет НОД-тест к оставшимся коэффициентам. Этот процесс продолжается до тех пор, пока не будет доказано, что уравнение является не допустимым или нет больше переменных, которые могут быть перемещены. Подробное теоретическое объяснение I-теста и некоторые преобразования интервальных уравнений описаны в [1].

*Многомерный I-тест [20]*

I-тест является эффективным и более точным методом анализа зависимостей по данным для одномерных массивов по сравнению с тестом Банержи. При анализе многомерных массивов λ-тест является эффективным тестом, но λ-тест дает ответ о существование вещественных решений системы уравнений зависимости. Многомерный I-тест представляет собой комбинацию данных двух тестов, следовательно, он дает более точный ответ о существовании целочисленных значений решений системы.

*λ-тест*

λ-тест [19] предназначен для *сцепленных* и *многомерных* ссылок массивов. Тест решает систему уравнений (1) и неравенств (2) и определяет, имеет ли данная система действительные решения.

Геометрически каждое линейное уравнение в (1) представляет собой гиперплоскость $\pi$ в пространстве $\mathbf{R^n}$. Пересечение **m** гиперплоскостей — $\boldsymbol{S}$ соответствует общим решениям системы (1). Очевидно, если $\boldsymbol{S}$ пусто, то никакой зависимости по данным нет. Границы циклов (2) соответствуют ограниченному выпуклому множеству **V** в $\mathbf{R^n}$. Уравнение имеет действительное решение, удовлетворяющее границам циклов и направлениям зависимостей, тогда и только тогда, когда соответствующая уравнению гиперплоскость $\pi$ пересекается с **V**. Тестирование «индекс-за-индексом» определяет, пересекается ли каждая гиперплоскость $\pi$ с **V**. Необходимо определить, пересекается ли **S** с **V**. Если из всех гиперплоскостей найдется такая гиперплоскость, которая не пересекает **V**, то очевидно **S** не может пересекаться с **V**. Однако, даже если каждая гиперплоскость из (1) пересекает **V**, существует вероятность того, что **S** не пересечет **V**. Если можно найти новую гиперплоскость, которая содержит **S**, но не пересекает **V**, то это дока-

зывает, что **S** и **V** не пересекаются. Имеется бесконечное число таких гиперплоскостей. Задача λ-теста — исследовать по мере необходимости некоторое количество гиперплоскостей, для определения пересечения **S** и **V**. В общем случае λ-тест генерирует $C_n^{m-1}$ таких гиперплоскостей, которые являются линейными комбинациями уравнений (1) и называются λ-плоскостями. Чтобы определить, пересекается ли каждая λ-плоскость с **V**, применяется тест Банержи для каждой λ-плоскости. Если хотя бы одна из λ-плоскостей не пересекает **V**, тогда зависимости по данным нет. Если каждая λ-плоскость пересекает **V**, то λ-тест принимает решение о возможном существовании зависимости.

*IR-тест*

IR-тест [20] является точным тестом, но он неприменим, когда значения границ циклов неизвестны или не являются константами. IR-тест находит целочисленные решения уравнения зависимости путем сокращения интервала решений переменных с многократным проектированием. Как только эффективный интервал решений какой-нибудь переменной сжимается к пустому, то уравнение зависимости, содержащее данную переменную, не имеет целочисленного решения.

*Модифицированный λ-тест*

В *модифицированном λ-тесте* [21] λ-тест интегрирован с точным **IR**-тестом, благодаря чему при анализе зависимостей многомерных массивов были получены более точные результаты.

*Модифицированный λ-тест* с помощью λ-теста генерирует множество линейных комбинаций гиперплоскостей (см. λ-тест). К сгенерированным гиперплоскостям применяется **IR**-тест для нахождения гиперплоскости из данного множества, которая не имеет целочисленных точек пересечения с **V**.

Идея нашей стратегии опирается на следующие научные факты и результаты.

### *1.2.4. Случаи, повышающие точность тестов на зависимость*

Точно определяющие методы: Омега-тест, Power-тест, алгоритм Майдана и др. используют линейные и целочисленные методы для решения диофантовых уравнений, например, метод Фурье—Моцкина, Симплекс метод и др., которые не эффективны на практике. В экспериментальных результатах Р. Триоле [22] показано, что по сравнению с более простыми

методами метод исключения переменных Фурье—Моцкина выполняется в 22–28 раз дольше.

Одним из стандартных и распространенных тестов на зависимость является тест Банержи. Он является приближенным тестом и принимает во внимание границы циклов. Эффективность и полноценность теста Банержи при опровержении зависимостей делают его одним из самых используемых тестов в распараллеливающих компиляторах

В исследованиях [16, 19, 5] показано, что если коэффициенты линейного уравнения удовлетворяют некоторым условиям, то тест Банержи становится точным тестом. Банержи показал, что его неравенства точны, если все коэффициенты индексных переменных равны 1, 0, или -1 [16]. Ли и др. [19] показали, что неравенства Банержи точны, если коэффициент одной индексной переменной $|a_k|=1$ и $|a_i| \le (U_i - L_i)$, где $i=1,\ldots,k-1,k+1,\ldots,n$.

Клапфолз (Klappholz) и др. [5] доказали, что неравенства Банержи точны, если после упорядочения коэффициентов индексных переменных $|a_1| \le |a_2| \le \ldots \le |a_n|$, коэффициент индексной переменной $|a_1| = 1$, и для каждого $j$ выполняется следующее условие: $\left|a_j\right| \le 1 + \sum_{k=1}^{j-1} \left|a_k\right| (U_k - L_k)$, $2 \le j \le n$.

## 2. ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ РЕЗУЛЬТАТОВ

Данный раздел посвящен анализу экспериментальных результатов, подтверждающих эффективность и корректность новой стратегии тестирования.

### 2.1. Системная среда

Эксперимент проведен с использованием инструмента **Petit V1.2** [10], разработанного в Мэрилендском университете как расширенный вариант инструмента ***tiny*** [11] и с использованием системы **SUIF** [12], разработанной в Стенфордском университете.

*Petit* является исследовательским инструментом реструктурирования программ. Он поддерживает совокупность библиотек и фундаментальных операций для анализа зависимостей по данным и реструктурирования программ. В нем в качестве алгоритмов анализа зависимостей по данным вне-

дрены четыре теста: Омега-тест, тест Банержи, Эпсилон-тест и Омега-Эпсилон-тест.

*SUIF* представляет собой инфраструктуру для исследований в области распараллеливающих и оптимизирующих компиляторов. Как сказано выше, в качестве библиотеки зависимости в системе служит алгоритм Майдана.

Обе программы были установлены на персональном компьютере с процессором AMD Athlon XP 1700+ с операционной системой Debian GNU Linux. Для эксперимента использованы два вида данных. Первый — набор тестовых научных программ NASA и PERFECT Club benchmarks [13], где каждая программа включает от 500 до 18000 строк (см. Таблицу 1). Второй вид — набор из 16 циклов, собранный из работ, аналогичных нашей [4, 6, 8, 9, 14, 15]. Все циклы являются специальными примерами и созданы для демонстрации мощности некоторых тестов на зависимость по данным.

Таблица 1

**Статистические характеристики эталонных тестовых программ**

| № | Эталонные тестовые программы | Количество строк программ | Количество подпрограмм | Количество DO-циклов | Количество ссылок на массивы |
|---|---|---|---|---|---|
| 1 | **LGSI** | 2815 | 36 | 161 | 6389 |
| 2 | **LWSI** | 1430 | 17 | 56 | 906 |
| 3 | **SDSI** | 8446 | 80 | 259 | 658 |
| 4 | **TISI** | 579 | 8 | 78 | 84 |
| 5 | **btrix** | 159 | 1 | 14 | 488 |
| 6 | **cholsky** | 53 | 1 | 18 | 70 |
| 7 | **gmtry** | 117 | 1 | 14 | 369 |
| 8 | **gosser** | 19 | 2 | 5 | 7 |
| | **Всего** | **13618** | **146** | **605** | **8971** |

В табл. 1 приведены статистические характеристики эталонных тестовых программ. Отметим, что эти программы специально были собраны для тестирования методов разных распараллеливающих и оптимизирующих компиляторов. Так как объектом распараллеливания служат большие научные пакеты прикладных программ, написанных на последовательных языках типа Фортран, каждая из этих программ решает задачу прикладной физики, математики, химии и др. В столбцах таблицы отражены названия, количество строк, количество подпрограмм, количество DO-циклов (for-циклы) каждой программы. При распараллеливании последовательных про-

грамм главным источником потенциального параллелизма, как правило, служит гнездо DO-циклов. В последнем столбце представлено количество ссылок на массивы.

### 2.2. Сравнение результатов

С помощью пакетов basesuif 1.3.0.1, suifbuilder 1.3.0.1, baseparsuif 1.3.0.1 и suifcookbook 1.3.0.1 системы SUIF [12] собраны два анализатора зависимостей по данным. Соответственно первый основан на алгоритме Майдана, а на втором внедрена новая стратегия. Каждый анализатор принимает на входе преобразованный на SUIF формат (с помощью scc драйвера) *.spd файл последовательной программы, а на выходе дает информацию о всех зависимостях по данным в данной программе. При экспериментальном сравнении результатов рассматривались только *истинные (потоковые) циклически порожденные зависимости по данным*.

Таблица 2

**Сравнение результатов**

| № | Эталонные тестовые программы | Всего обрщ. на истин. завис. | Кол-во разрушенных зависимостей | | | |
|---|---|---|---|---|---|---|
| | | | Алгоритм Майдана | | Новая стратегия | |
| 1 | **LGSI** | 6168 | 5546 | 89,92% | 5546 | 89,92% |
| 2 | **LWSI** | 795 | 102 | 12,83% | 102 | 12,83% |
| 3 | **SDSI** | 417 | 154 | 36,93% | 149 | 35,73% |
| 4 | **TISI** | 52 | 0 | 0,00% | 0 | 0,00% |
| 5 | **btrix** | 450 | 208 | 46,22% | 208 | 46,22% |
| 6 | **cholsky** | 61 | 3 | 4,92% | 0 | 0,00% |
| 7 | **gmtry** | 258 | 125 | 48,45% | 119 | 46,12% |
| 8 | **gosser** | 5 | 0 | 0,00% | 0 | 0,00% |
| | **Всего** | **8206** | **6138** | **74,80%** | **6124** | **74,63%** |

Результаты каждого анализатора зависимостей по данным для эталонных тестовых программ показаны в табл. 2. Третья колонка табл. 2 представляет общее количество обращений на предмет наличия потоковой зависимости по данным, здесь тесты должны разрушать зависимости, если самом деле не существует потоковой зависимости по данным. Следовательно, в следующих колонках таблицы показано, на сколько процентов удалось

разрушить зависимости с помощью алгоритма Майдана и с алгоритма новой стратегии соответственно.

Алгоритм Майдана считается дорогим и точным тестом, т.к. он использует метод исключения Фурье—Моцкина, имеющий теоретически экспоненциальную временную сложность. Из табл. 2. видно, что результаты нового алгоритма очень близки к результатам точных тестов, хотя предложенный алгоритм имеет полиномиальную временную сложность в наихудшем случае.

### 2.3. Сравнение результатов на экспериментальных примерах

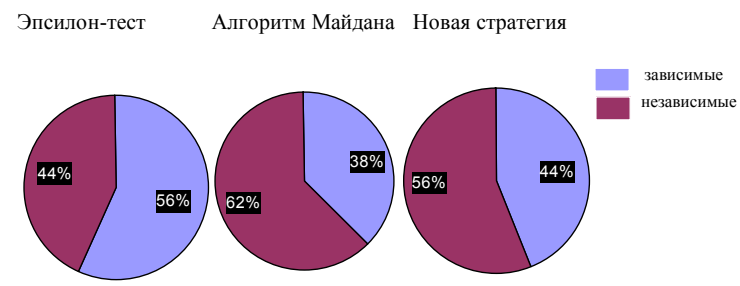
Для повышения качества эксперимента был собран набор из 16 циклов из работ аналогичных нашей: Maydan [8], Wolfe [14], Goff [6], Pugh [15], Yang [9] и Shen [4]. Все циклы являются специальными примерами и созданы для демонстрации мощности некоторых тестов на зависимость по данным. В данных примерах индексные выражения более сложны, чем реальные программы. Статистические данные показаны в табл. 3.

Таблица 3

**Характеристики циклов в экспериментальных примерах**

| Пример / количество | 1-мерные | 2-мерные | **Всего** |
|---|---|---|---|
| Maydan | 4 | 1 | 5 |
| Wolfe | 0 | 4 | 4 |
| Goff | 0 | 1 | 1 |
| Pugh | 1 | 0 | 1 |
| Yang | 2 | 1 | 3 |
| Shen | 2 | 0 | 2 |
| **Всего** | 9 | 7 | **16** |

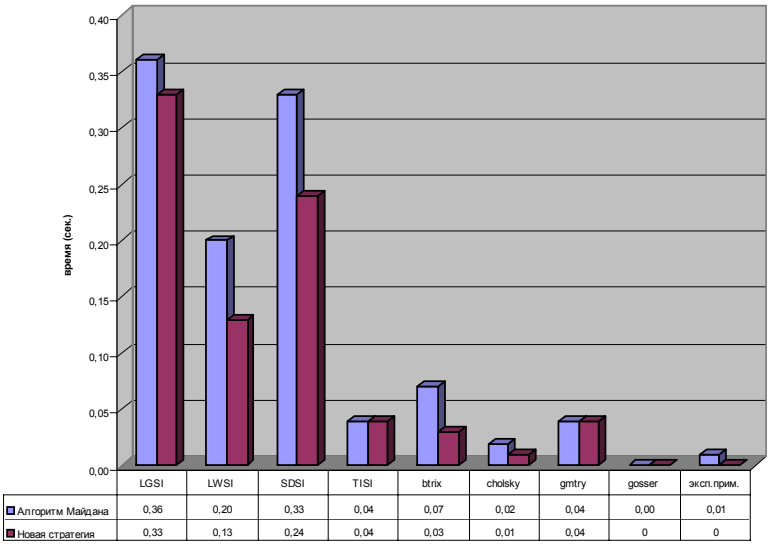
Общие статистические характеристики экспериментальных примеров: количество строк — 66, количество DO-циклов — 26, количество ссылок на массивы — 40. В этом случае мы сравнивали результаты следующих алгоритмов: Эпсилон-тест, алгоритм Майдана и новая стратегия тестирования. С помощью инструмента Petit к экспериментальным примерам применялся Эпсилон-тест. Соответственно для 40 пар ссылок массивов получены следующие данные о зависимости по данным (см. рис 2).




| Всего ссылок | Эпсилон тест | Алгоритм Майдана | Новая стратегия |
|---|---|---|---|
| 16 | 7 | 10 | 9 |

*Рис. 2.* Сравнение результатов на экспериментальных примерах

## 2.4. Время выполнения




| | LGSI | LWSI | SDSI | TISI | btrix | cholsky | gmtry | gosser | эксп.прим. |
|---|---|---|---|---|---|---|---|---|---|
| Алгоритм Майдана | 0,36 | 0,20 | 0,33 | 0,04 | 0,07 | 0,02 | 0,04 | 0,00 | 0,01 |
| Новая стратегия | 0,33 | 0,13 | 0,24 | 0,04 | 0,03 | 0,01 | 0,04 | 0 | 0 |

*Рис. 3.* Время выполнения

Для определения того, какой метод требует больше времени исполнения, использован *GNU* профилятор ***`gprof'*** [23] операционной системы Linux с периодом отсчета 0,01 сек. Чтобы снизить статистические неточности, каждый алгоритм зависимости по данным выполнен 100 раз для различных эталонных тестов, и взято их усредненное значение (см. рис. 3). В общем случае алгоритм Майдана требовал времени на 25% больше, чем алгоритм новой стратегии.

### 2.5. Статистические данные *Новой стратегии*

По итогам эксперимента были анализированы статистические данные новой стратегии (см. табл. 4). Результаты еще раз показывают целесообразность и эффективность методов, предложенных в предыдущей главе. В большинстве случаев зависимости по данным выявляются с помощью простых тестов: константный тест (ZIV-тест), SIV-тест, Банержи тест и λ-тест. Помощь более сложных тестов (I-тест, IR-тест, Многомерный I-тест и Модифицированный λ-тест) требовалась в незначительных случаях. Это достигается с помощью алгоритмов, уточняющих ответы теста Банержи. Так, в 361 случае с помощью Банержи теста в 54 случаях опровергнуты зависимости по данным, а в 301 случае алгоритм проверки коэффициентов доказывает существование зависимости. Это позволяет сократить общее время выполнения новой стратегии, так как после положительного ответа алгоритма проверки исключается выполнение последовательности более сложных тестов (I-тест, IR-тест или Многомерный I-тест, Модифицированный λ-тест). В 71 случае с помощью λ-теста опровергнуты зависимости по данным в 2 случаях, а 67 случае алгоритм проверки коэффициентов доказывает существование зависимости.

## 3. ЗАКЛЮЧЕНИЕ

Анализ зависимостей по данным является фундаментальным компонентом в распараллеливающих компиляторах. В данной работе предложен новый эффективный алгоритм анализа зависимостей по данным, а также проведено экспериментальное сравнение результатов предложенного метода с наиболее известными стратегиями тестирования анализа зависимостей по данным, такими как Эпсилон-тест и алгоритм Майдана.

Таблица 4

**Статистические данные Новой стратегии**

| тесты на зависимость | | LGSI | LWSI | SDSI | TISI | btrix | cholsky | gmtry | gosser | прак.дан. | Всего |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Константный-тест** | обращ. | 5862 | 42 | 185 | 0 | 208 | 2 | 230 | 0 | 0 | **6529** |
| | независ. | 5455 | 18 | 149 | 0 | 208 | 0 | 119 | 0 | 0 | **5949** |
| **НОД-тест** | обращ | 4 | 163 | 82 | 20 | 198 | 56 | 19 | 5 | 2 | **549** |
| | незав. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | **1** |
| **SIV-тест** | обращ | 266 | 100 | 11 | 0 | 0 | 0 | 9 | 0 | 5 | **391** |
| | незав. | 91 | 31 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | **126** |
| **Тест Банержи** | обращ | 1 | 319 | 2 | 0 | 30 | 0 | 1 | 0 | 8 | **361** |
| | незав. | 0 | 53 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | **54** |
| | проверка | 1 | 266 | 2 | 0 | 30 | 0 | 1 | 0 | 1 | **301** |
| **I-тест** | обращ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 | **8** |
| | незав. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0** |
| **IR-тест** | обращ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | **1** |
| | незав. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | **1** |
| **Лямбда-тест** | обращ | 0 | 0 | 64 | 0 | 0 | 0 | 0 | 0 | 7 | **71** |
| | незав. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | **2** |
| | проверка | 0 | 0 | 64 | 0 | 0 | 0 | 0 | 0 | 3 | **67** |
| **Многомерный I-тест** | обращ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | **2** |
| | незав. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0** |
| **Модифицированный Лямбда-тест** | обращ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | **2** |
| | незав. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | **1** |

Экспериментальное исследование показывает, что при анализе зависимостей по данным на эталонных тестовых программах NASA и PERFECT Club benchmarks, результаты нового алгоритма очень близки к результатам алгоритма Майдана. Хотя алгоритм Майдана считается дорогим и точным тестом, а предложенный новый алгоритм имеет полиномиальную вре-

менную сложность в наихудшем случае. В общем случае алгоритм Майдана требовал времени на 25% больше, чем алгоритм новой стратегии.

Сравнение результатов на экспериментальных примерах показало, что новый алгоритм выявляет примерно на 12% больше ложных зависимостей, чем аналогичные приближенные алгоритмы (Эпсилон-тест), и только примерно на 6% уступает алгоритму Майдана.

Все результаты, полученные экспериментальным путем, еще раз показывают целесообразность и эффективность новой стратегии тестирования, предложенной в данной работе.

Практическим результатом данной работы является алгоритм, который может быть использован в блоке анализа зависимостей по данным в проектируемой системе быстрого прототипирования компилятора.

## СПИСОК ЛИТЕРАТУРЫ


1. **Евстигнеев В.А.** Анализ зависимостей: состояние проблемы // Системная информатика: Сб. науч. тр. / Ин-т систем информатики СО РАН. — Новосибирск: Наука, 2000. — Вып. 7. — С. 112–173.
2. **Евстигнеев В.А., Арапбаев Р.Н., Осмонов Р.А.** Анализ зависимостей: основные тесты на зависимость по данным // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2007. — Т. 10, № 3. — С. 247–265.
3. **Арапбаев Р.Н., Осмонов Р.А.** Анализ зависимостей: новая стратегия тестирования // Тр. Междунар. конф. «Параллельные вычислительные технологии (ПаВТ'2007)». — Челябинск: Ид-во ЮУрГУ, 2007. — Т. 2. — С. 16–27.
4. **Shen, Z., Li, Z., Yew, P.-C.** An empirical study of Fortran programs for parallelizing compilers // IEEE Transaction on Parallel and Distributed Systems. — 1992. — Vol. 1 (1). — P. 356–364.
5. **Psarris K.** Program analysis techniques for transforming programs for parallel execution // Parallel Computing. — 2002. — Vol. 28. — P. 455–469.
6. **Goff G., Kennedy K., Tseng C.** Practical Dependence Testing // Proc. of the ACM SIGPLAN 91 Conf on Programming Language Design and Implementation, June 1991. — P. 15–29.
7. **Pugh W., Speismean T**. On Fast Array Data Dependence Tests. — Univ. of Maryland, College Park, January 3, 1999. — http://citeseer.ist.psu.edu/43683.htm
8. **Maydan D., Hennessy J., Lam M.** Efficient and Exact Data Dependence Analysis // Proc. of the SIGPLAN Conf. on Programming Language Design and Implementation, June 1991. — P. 1–14.
9. **Yang C.-T., Tseng S.-S., Shih W.-C.** The K Test: an Exact and Efficient Knowledge-based Data Dependence Testing Method for Parallelizing Compilers // Proc. Natl. Sci. Counc. ROC(A). — 2000. — Vol. 24, N 5. — P. 362–372.

10. **Kelly W., Maslov V., Pugh W., et al.** Petit: a tool for analyzing and transforming array calculations. — Dept. of Computer Science, University of Maryland, College Park, April 1996. — http://www.cs.umd.edu/projects/omega/index.html
11. **Wolfe M.** The Tiny Loop Restructuring Research Tool // Proc. of the 1991 Internat. Conf. on Parallel Processing, St Charles, IL, August 1991.
12. **Wilson R.P., French R.S., Wilson C.S. a.o.** SUIF: An infrastructure for research on parallelizing and optimizing compilers // SIGPLAN Not. —1994. — Vol. 29, N 12. — P. 31–37.
13. **Berry M. et al.** The PERFECT Club benchmarks: effective performance evaluation of supercomputers. Technical Report UIUCSRD Rep. No. 827, University of Illinois Urbana-Champaign, 1989, 48 p.
14. **Wolfe M., Tseng C.** The Power Test for Data Dependence // IEEE Transactions on Parallel and Distributed Systems. September 1992.
15. **Pugh W.** The Omega test: a fast and practical integer programming algorithm for dependence analysis // Communs. of the ACM. — 1992. — Vol. 35(8). — P.102–114.
16. **Banerjee U.** Data dependence in ordinary programs. — Urbana, 1976. — (Univ.III., Technical Rep. 76-837).
17. **Kong X., Klappholz D., Pssaris K.** The I Test: An Improved Dependence Test for Automatic Parallelization and Vectorization // IEEE Transactions on Parallel and Distributed Systems. — 1991. — Vol. 2(1). — P. 342–349.
18. **Huang T.-C., Yang C.-M.** Data dependence analysis for array references // J. of Systems and Software. — 2000. — Vol. 52. — P. 55–65.
19. **Li Z., Yew P.-C., Zhu C.-Q.** An efficient data dependence analysis for parallelizing compilers. // IEEE Transaction on Parallel and Distributed Systems. — 1990. — Vol. 1(1). — P. 26–34.
20. **Chang W.-L., Chu C.-P., Wu J.** A multi-dimensional version of the I test // Parallel Computing. — 2001. — Vol. 27. — P. 1783–1799.
21. **Арапбаев Р.Н., Осмонов Р.А.** Анализ зависимостей по данным для многомерных массивов на базе модифицированного λ-теста // Проблемы интеллектуализации качества систем информатики. — Новосибирск: ИСИ СО РАН, 2006. — С. 7–23.
22. **Triolet R.,** Interprocedural analysis for program restructuring with PARAFRASE. — Urbana, 1985 — (Tech. Rep. / Univ. III. CSRD; N 538).
23. **Fenlason and Stallman**, GNU gprof, The GNU Profiler. — http:// www.gnu.org/manual/gprof-2.9.1/html_chapter/gprof_toc.html