УДК 004.272, 004.021 **DOI: 10.14529/cmse250202**

ПОИСК ОПТИМАЛЬНЫХ ВЕСОВ ДЛЯ ФУНКЦИИ ЯДРА АКУШСКОГО

© 2025 В.В. Луценко, Д.Е. Горлачев, Н.М. Мирный, М.Г. Бабенко

Северо-Кавказский федеральный университет
(355017 Ставрополь, ул. Пушкина, д. 1)
E-mail: officialvladlutsenko@gmail.com, dimagorlacev12@gmail.com,
mirnyjn@gmail.com, mgbabenko@ncfu.ru

Поступила в редакцию: 07.03.2025

Современные вычислительные задачи, связанные с обработкой больших чисел, требуют не только высокой точности, но и значительной скорости. В этом контексте использование системы остаточных классов предлагает подход к параллельной обработке больших данных, применяемый в криптографии, обработке сигналов и искусственных нейронных сетях. Несмотря на преимущества системы остаточных классов, ее распространение замедленно в связи с вычислительной сложностью так называемых немодульных операций системы остаточных классов. Одним из универсальных инструментов для реализации немодульных операций является функция ядра Акушского. В работе исследуется функция ядра Акушского как инструмент для определения позиционной характеристики числа в системе остаточных классов. Для поиска оптимальных весов функции ядра предложено применение метода Монте-Карло и генетического алгоритма. Экспериментальные результаты демонстрируют, что генетический алгоритм обеспечивает более стабильные результаты при увеличении количества модулей, в то время как метод Монте-Карло эффективен на малых размерностях. Генетический алгоритм в среднем на 38% быстрее метода Монте-Карло, что делает его предпочтительным выбором. Дополнительно проведено сравнение времени вычисления функции ядра с оптимальными весами и функции Пирло. Результаты показали, что функция ядра с оптимальными весами в среднем на 14% быстрее функции Пирло.

Ключевые слова: система остаточных классов, высокопроизводительные вычисления, функция ядра Акушского, метод Монте-Карло, генетический алгоритм.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Луценко В.В., Горлачев Д.Е., Мирный Н.М., Бабенко М.Г. Поиск оптимальных весов для функции ядра Акушского // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2025. Т. 14, № 2. С. 26–41. DOI: 10.14529/cmse250202.

Введение

Система остаточных классов (СОК) представляет собой математический аппарат, который находит широкое применение в различных областях, таких как криптография [1], обработка сигналов [2], вычислительная техника [3] и теория чисел [4]. Основная идея СОК заключается в представлении чисел в виде набора остатков от деления на взаимно простые модули, что позволяет упростить арифметические операции и повысить эффективность вычислений. В отличие от традиционных позиционных систем счисления, СОК предлагает параллельный подход к обработке данных, что особенно полезно в задачах, требующих высокой скорости и точности.

Исторически система остаточных классов восходит к Китайской теореме об остатках (КТО). В последние десятилетия интерес к СОК возрос благодаря ее применению в квантовых вычислениях [5], нейронных сетях [6, 7].

Несмотря на преимущества СОК у нее существуют недостатки в виде так называемых немодульных операций. К данным операциям относятся: обратное преобразование числа,

Таблица 1. Представление чисел для СОК с базисом $\{3, 4\}$

$0 \xrightarrow{\text{СОК}} (0,0)$	$1 \xrightarrow{\text{СОК}} (1,1)$	$2 \xrightarrow{\text{СОК}} (2,2)$	$3 \xrightarrow{\text{СОК}} (0,3)$
$4 \xrightarrow{\text{СОК}} (1,0)$	$5 \xrightarrow{\text{СОК}} (2,1)$	$6 \xrightarrow{\text{СОК}} (0,2)$	$7 \xrightarrow{\text{СОК}} (1,3)$
$8 \xrightarrow{\text{СОК}} (2,0)$	$9 \xrightarrow{\text{СОК}} (0,1)$	$10 \xrightarrow{\text{СОК}} (1,2)$	$11 \xrightarrow{\text{СОК}} (2,3)$

расширение базиса, сравнение чисел, определение знака числа, обнаружение переполнения, общее деление, масштабирование и другие. Для выполнения немодульных операций необходимо знать так называемую позиционную характеристику (ПХ) числа в СОК. Универсальным инструментом для определения ПХ является так называемая функция ядра Акушского [8]. Функция ядра Акушского может эффективно применяться для обратного преобразования в позиционную систему счисления [9], определения знака числа [10], определение четности числа [11], общего деления в СОК [12].

Однако несмотря на все исследования, как и все остальные методы поиска ПХ, вычисление значения функции ядра остается вычислительно сложной задачей. С целью снизить вычислительную сложность нахождения значения функции ядра мы предлагаем использовать оптимальные веса для функции ядра поиск которых осуществлять с использованием метода Монте-Карло или генетического алгоритма.

Статья имеет следующую структуру. В разделе 1 рассматривается система остаточных классов. В разделе 2 представлены основы функции ядра Акушского. В разделе 3 представлены алгоритмы поиска оптимальных весов для функции ядра Акушского. Наконец, в разделе 4 проведен анализ эффективности предложенных методов. В заключении суммируются полученные результаты.

1. Основы системы остаточных классов

Определение 1. Базисом системы остаточных классов называется множество модулей $\{p_1, p_2, ..., p_n\}$, где каждый модуль $p_i \geqslant 2$ $(i = 1, 2, \ldots, n)$ и $\gcd(p_i, p_j) = 1$. Здесь и далее по тексту gcd обозначает наибольший общий делитель (англ. greatest common divisor). По умолчанию модули упорядочены по возрастанию, то есть $p_1 < p_2 < \cdots < p_n$.

Определение 2. Произведение модулей $P = \prod_{i=1}^{n} p_i$ называется динамическим диапазоном системы остаточных классов.

Определение 3. Целое число $X \in [0, P)$ представимо в виде n-мерного вектора, составленного из наименьших неотрицательных остатков от деления соответствующего числа на p_i:

$$X = (x_1, x_2, \ldots, x_n),$$

где $x_i \equiv X \mod p_i$, что также обозначается как $x_i = |X|_{p_i}$.

В табл. 1 представлены соответствия чисел из позиционной системы счисления и системы остаточных классов для базиса $\{3, 4\}$.

Система остаточных классов обладает особенностью выполнять операции сложения, вычитания и умножения параллельно и независимо для каждого модуля. Для чисел $X = (x_1, x_2, \ldots, x_n)$ и $Y = (y_1, y_2, \ldots, y_n)$ выразим арифметику следующим образом:

$$X * Y = (x_1 * y_1, x_2 * y_2, \ldots, x_n * y_n), \text{где } * \in \{+, -, \times\}. \quad (1)$$

В примере 1 представлено сложение двух чисел в СОК.

Пример 1. Сложим два числа $X = 5$ и $Y = 4$ в базисе $\{3, 4\}$. Их представление в заданном базисе указано в табл. 1. Воспользуемся уравнением (1) для сложения:

$$X + Y = (|2 + 1|_3 , |1 + 0|_4) = (0, 1) .$$

Отсутствие переносов при выполнении модульных операций — одно из ключевых преимуществ арифметики в СОК. Это означает, что результаты операций сложения и умножения остаются в пределах заданного диапазона значений для каждого модуля. Это значительно упрощает и ускоряет процесс выполнения операций сложения и умножения по сравнению с позиционными системами счисления.

Преимущество арифметики с остатками особенно ярко проявляется при выполнении вычислительных процедур, состоящих из последовательности модулярных арифметических операций. При этом можно выбрать набор модулей таким образом, чтобы конечные результаты всегда находились в допустимом диапазоне значений для любых входных операндов. Промежуточные результаты могут выходить за границы диапазона, но это не представляет проблемы.

Несмотря на преимущества системы остаточных классов в некоторых аспектах, она имеет ряд существенных ограничений в других. Например, для чисел $(0, 3)$ и $(0, 2)$ из табл. 1, невозможно сразу определить, какое из них больше, что требует использования специальных методов для вычисления позиционной характеристики.

Для того чтобы восстановить позиционное представление числа в СОК, можно воспользоваться Китайской теоремой об остатках [13].

Теорема 1. Пусть $\{p_1, p_2, \ldots, p_n\}$ есть некоторые натуральные взаимно простые числа и $P = \prod_{i=1}^{n} p_i$. Любое число X, такое что $0 \leqslant X < P$, может быть однозначно представлено в виде последовательности $(x_1, x_2, \ldots, x_n)$, где $x_i = X \bmod p_i$, при этом

$$X = \left| \sum_{i=1}^{n} P_i \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_P , \tag{2}$$

где $P_i = \frac{P}{p_i}$, $\left| P_i^{-1} \right|_{p_i}$ мультипликативная инверсия P_i по модулю p_i.

Определение 4. Значения $B_i = \left| P_i^{-1} \right|_{p_i} \cdot P_i$ называются ортогональными базисами системы остаточных классов.

Пример 2. Возьмем СОК с набором модулей $\{3, 4\}$. Рассмотрим процесс перевода числа $X = (0, 2)$ в позиционную систему счисления.

Найдем значения P_i и $\left| P_i^{-1} \right|_{p_i}$:

$$P_1 = \frac{P}{p_1} = 4, \quad P_2 = \frac{P}{p_2} = 3.$$
$$\left| P_1^{-1} \right|_{p_1} = 1, \quad \left| P_2^{-1} \right|_{p_2} = 3.$$

По формуле 2 найдем X:

$$X = |4 \cdot 1 \cdot 0 + 3 \cdot 3 \cdot 2|_{12} = 6.$$

Таким образом, получено число в позиционной системе.

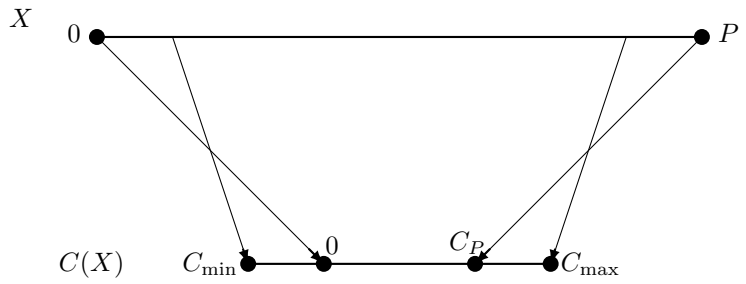

Рис. 1. Отображение $0 \leqslant X < P$ на $C_{min} \leqslant C(X) < C_{max}$

Немодульные операции, такие как обратное преобразование числа, общее деление, сравнение чисел, определение знака числа являются вычислительно сложными и требуют большого количества вычислений, что ограничивает применение СОК и ее использование для быстрых вычислений.

2. Функция ядра Акушского

Исследования, проведенные Акушским И.Я., Бурцевым В.М. и Паком И.Т. [14], были направлены на нахождение позиционной характеристики числа в СОК. В результате этих исследований была предложена новая математическая конструкция, получившая название функции ядра Акушского. Функция ядра Акушского определяется следующей формулой:

$$C\left(X\right) = \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{X}{p_i} \right\rfloor, \tag{3}$$

где целые числа w_i выступают в роли постоянных величин, которые определяются выбором точки для интерполяции. Константы w_i задают вес каждого из частных $\left\lfloor \frac{X}{p_i} \right\rfloor$ в формуле (3), тем самым задавая функцию ядра и придавая ей различные свойства.

Подставив $X = P$ в (3), получим:

$$C\left(P\right) = C_P = \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{P}{p_i} \right\rfloor = \sum_{i=1}^{n} w_i \cdot P_i. \tag{4}$$

Разделив $C\left(P\right)$ на P, получим:

$$\frac{C\left(P\right)}{P} = \sum_{i=1}^{n} \frac{w_i}{p_i}. \tag{5}$$

Отсюда следуют, что некоторые значения w_i должны быть отрицательными, чтобы получить малые значения $C\left(P\right)$. Подставив (5) в (3), получим:

$$C\left(X\right) = X \cdot \frac{C\left(P\right)}{P} - \sum_{i=1}^{n} \frac{x_i \cdot w_i}{p_i} = X \cdot \sum_{i=1}^{n} \frac{w_i}{p_i} - \sum_{i=1}^{n} \frac{x_i \cdot w_i}{p_i}. \tag{6}$$

Используя функцию ядра можно получить информацию о позиционной характеристики числа как показано на рис. 1.

Уравнение (6) указывает, что график $C\left(X\right)$ относительно X должен быть прямой линией с наклоном $C(P)/P$ с некоторой нелинейностью. Величина нелинейности определяется

величиной весов, в свою очередь связанной с конкретным значением $C(P)$ для заданного набора модулей СОК.

Учитывая, что $P_j \equiv 0 \mod p_i$ для любого $i \neq j$, веса w_i функции $C(X)$ могут быть определены соотношением:

$$w_i \equiv \left| C(P) \cdot P_i^{-1} \right|_{p_i}. \tag{7}$$

Таким образом, веса могут быть определены после выбора $C(P)$, но с условием выполнения (4).

Пример 3. Рассмотрим систему с модулями $p_1 = 3$, $p_2 = 4$ и весами $w_1 = 1$, $w_2 = 2$. Динамический диапазон $P = 12$. Функция ядра от динамического диапазона равна $C(P) = 10$. Для данных параметров график функции ядра изображена на рис. 2.

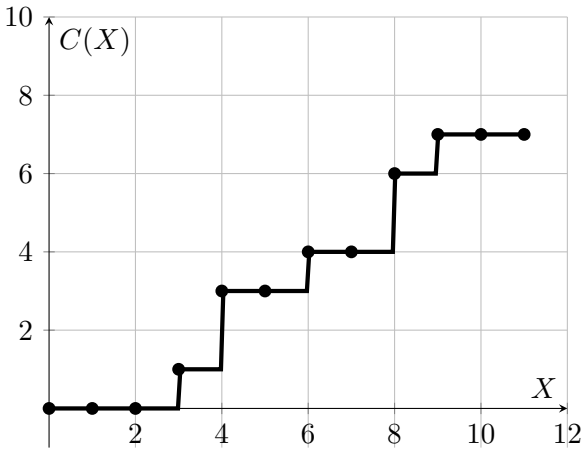

Рис. 2. График функции ядра с $w_1 = 1$, $w_2 = 2$ для СОК с основаниями $p_1 = 3$, $p_2 = 4$

Отметим, что (3) плохо подходит для вычисления значений функции ядра в практических приложениях, так как необходимо знать позиционное представление X. Применяя КТО, возможно модифицировать подход к расчету значения функции ядра для X, основываясь на ее остатках.

Значение функции ядра $C(X)$ с весами $w_1, w_2, \ldots, w_n$, при условии $0 \leqslant C(X) < C_P$, $X \in [0, P)$, можно вычислить с использованием формулы

$$C(X) \equiv \left| \sum_{i=1}^{n} k_i \cdot x_i \right|_{C_P}, \tag{8}$$

где $k_i = C(B_i)$.

Однако в общем случае при определенных w_i, $C(X)$ может не удовлетворять условию $0 \leqslant C(X) < C_P$ для $X \in [0, P)$, тогда возникает проблема так называемых критических ядер, которая ограничивает возможность использовать формулу (8). В данном случае функции ядра будут располагаться в некотором промежутке $[-\delta_1, C_P + \delta_2]$ (см. рис. 1).

В статье [8] рассмотрены эффективные методы, которые позволяют определить критические ядра.

В данной статье мы сосредоточимся на поиске оптимальных весов для функции ядра Акушского. Для этого введем следующее определение.

Определение 5. Будем называть функцию ядра Акушского дружественной для программной и аппаратной реализации, если $C(P) = C_P = 2^N$, где $C(P) = \sum_i^n w_i P_i$.

Теорема 2. Для множество модулей $\{p_1, p_2, \ldots, p_n\}$ где $p_i \geqslant 2$ и $\gcd(p_i, p_j) = 1$, $\forall i \neq j$, существует набор неотрицательных весов $\{w_1, w_2, \ldots, w_n\}$, таких, что

$$C(P) = \sum_{i=1}^{n} w_i \cdot P_i = 2^N.$$

Доказательство. Докажем теорему методом индукции.

1. База индукции ($n = 1$). Если $n = 1$, то у нас есть только один модуль $p_1 \geqslant 2$, тогда $P = p_1$ и $P_1 = \frac{P}{p_1} = 1$.

Тогда существует единственный вес $w_1 = 2^N$, который удовлетворяет условию $C(P) = 2^a$.

2. Индукционное предположение ($n = k$). Предположим, что теорема верна для $n = k$, то есть для любого множества модулей $\{p_1,\ p_2, \ldots,\ p_k\}$, тогда с учетом свойств КТО существуют такие неотрицательные веса $\{w_1, w_2, \ldots, w_k\}$, что

$$C(D_k) = \sum_{i=1}^{k} w_i \cdot P_i = 2^N,$$

где $D_k = \prod_{n=1}^{k} p_i$.

3. Шаг индукции ($n = k + 1$). Рассмотрим множество из $k + 1$ попарно взаимно простых модулей $\{p_1, p_2, \ldots, p_{k+1}\}$. Обозначим $D_{k+1} = D_k \cdot p_{k+1}$. Согласно предположению индукции, для множества модулей $\{p_1, p_2, \ldots, p_k\}$ существует набор неотрицательных весов $\{w_1, w_2, \ldots, w_k\}$, таких что $C(D_k) = 2^N$.

Теперь добавим новый модуль p_{k+1} и определим соответствующий вес w_{k+1}, такой что сумма

$$C(D_{k+1}) = C(D_k) + w_{k+1} \cdot P_{k+1} = 2^N.$$

Заметим, что данное условие выполняется, например при $w_{k+1} = 0$.

□

Приведем пример 4 работы с функцией ядра, когда она является дружественной.

Пример 4. Пусть дана система модулей $p_1 = 3$, $p_2 = 4$, $p_3 = 5$. Объем динамического диапазона $P = 3 \cdot 4 \cdot 5 = 60$. Покажем, что веса $w_1 = 1$, $w_2 = 0$, $w_3 = 1$ являются оптимальными для данной системы.

Найдем значения P_i:

$$P_1 = \frac{P}{p_1} = 20,\ P_2 = \frac{P}{p_2} = 15,\ P_3 = \frac{P}{p_3} = 12.$$

Далее по формуле (4) найдем диапазон функции ядра:

$$C(P) = 1 \cdot 20 + 0 \cdot 15 + 1 \cdot 12 = 32 = 2^5.$$

Значения ортогональных базисов B_i равны:

$$B_1 = 40,\ B_2 = 45,\ B_3 = 36.$$

Далее найдем коэффициенты $k_i = C(B_i)$:

$$k_1 = 21,\ k_2 = 24,\ k_3 = 19.$$

Найдем значение $C(X)$ для числа $(2, 3, 1)$:

$$C(X) = |21 \cdot 2 + 24 \cdot 3 + 19 \cdot 1|_{2^5} = |133|_{2^5} = |[10000101]_2|_{2^5} = [101]_2 = 5.$$

Таким образом, вычислительно сложную операцию взятия по модулю можно заменить срезом $N = 5$ старших битов.

В следующем разделе рассмотрим методы поиска оптимальных весов функции ядра, при использовании которых будет выполняться условие $C_P = 2^N$.

3. Методы подбора оптимальных весов для функции ядра

3.1. Метод Монте-Карло

Метод Монте-Карло — это группа численных методов, основанных на случайном моделировании (стохастическом подходе), которые применяются для решения различных математических, статистических и других задач [15]. Метод был разработан в середине двадцатого века, главным образом в контексте исследований, связанных с ядерной физикой, но быстро нашел применение во множестве других дисциплин. Метод основан на использовании случайных или псевдослучайных чисел. Они применяются для имитации поведения сложных систем или случайных процессов. Благодаря своей случайности, метод позволяет быстро находить решения в больших диапазонах. В работе [16] были описаны основные преимущества данного метода и подробное разъяснение в различных сферах его использования.

В статье [17] было предложено использование метода Монте-Карло для поиска весов функции ядра, которые увеличивают точность вычисления приближенного ранга числа в СОК. Данный метод может быть интегрирован в системе остаточных классов, для поиска оптимальных весов функции ядра Акушского, удовлетворяющих условию $C_P = 2^N$. Ниже представлен алгоритм 1, который реализует данный метод.

Метод Монте-Карло эффективен для небольших базисов, однако с увеличением количества модулей вероятность нахождения оптимальных весов падает, из-за этого время выполнения алгоритма может быть значительным. С целью ускорения подбора оптимальных весов предложено использовать генетический алгоритм.

3.2. Генетический алгоритм

Генетический алгоритм — это метод оптимизации и поиска, основанный на принципах естественного отбора и эволюции в биологии [18]. Генетические алгоритмы часто рассматриваются как оптимизаторы функций, хотя спектр задач, к которым применяются генетические алгоритмы, довольно широк. Метод является разновидностью эволюционных вычислений, таких как наследование, мутации, отбор и кроссинговер. Из работы [19] можно выделить следующую структуру:

Алгоритм 1: Метод Монте-Карло для поиска оптимальных весов функции ядра

Input: $\{p_1, p_2, \ldots, p_n\}$,
$max\_iterations$,
$weigh\_limit$
Data: $P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}, i = 1, 2, \ldots, n$
Output: $\{w_1, w_2, \ldots, w_i\}, N$ — если найдены оптимальные веса

```
for i := 1 to max_iterations do
    w_i = random(0, weight_limit)
    C_P = Σ_i^n w_i · P_i
    if C_P > 0 and (C_P ∧ (C_P − 1)) = 0 then
        N = log_2 C_P
        return w_i, N
```

- Ген — это структура данных, представляющая одно возможное решение задачи. Множество генов — популяция.
- Функция приспособленности оценивает качество генов. Она определяет, насколько решение соответствует заданным критериям.
- Селекция (selection) — выбор генов (родителей) для создания следующего поколения. Обычно выбираются гены с высоким значением функции приспособленности.
- Скрещивание (crossover) — объединение генетической информации двух родителей для создания одного или нескольких потомков.
- Мутация (mutation) — внесение случайных изменений в потомков для поддержания разнообразия и предотвращения застревания в локальных максимумах.

Такой алгоритм можно применить для поиска оптимальных весов. Для начала нужно будет задать начальную популяцию, путем генерации случайных весов. Оценивать приспособленность будем через степень двойки функции C_P. Чем меньше значение функции приспособленности, тем лучше данное решение.

Далее выбираются лучшие веса из текущей популяции, происходит этап селекции. Отбирается верхняя половина популяции — веса с наименьшими значениями функции приспособленности. Эти лучшие решения используются для создания потомков.

Создаются новые «потомки» путем рекомбинации генов (весов) двух родителей, выбирается точка скрещивания. Для поддержания генетического разнообразия потомки подвергаются мутации с определенной вероятностью и затем, новая популяция формируется из лучших индивидов текущего поколения и потомков, тем самым получая оптимальные веса. Ниже представлен псевдокод данного алгоритма.

4. Оценка производительности

Моделирование и вычислительные эксперименты проведены на компьютере, оснащенным процессором Intel Core i7-7700HQ с тактовой частотой 2.80 ГГц, 8 ГБ оперативной памяти DDR4 с частотой 1196 МГц и твердотельным накопителем емкостью 512 ГБ, работающий под управлением Windows 10 Home Edition, с использованием языка программирования высокого уровня Python.

Алгоритм 2: Генетический алгоритм для поиска оптимальных весов функции ядра

Input: $\{p_1, p_2, \ldots, p_n\}$,
$population\_size$,
$weight\_limit$
Data: $P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}, i = 1, 2, \ldots, n$
Output: $\{w_1, w_2, \ldots, w_i\}, N$ — если найдены оптимальные веса

$w_i = random(0, weight\_limit)$
// Оценка приспособленности:
$C_P = \sum_i^n w_i \cdot P_i$
if $C_P > 0$ **and** $(C_P \ \wedge \ (C_P - 1)) = 0$ **then**
    $N = \log_2 C_P$
    return w_i, N
// Селекция:
$n = population\_size$
$f_{w_i} = \frac{C(P)}{\sum\limits_{i=1}^{n} C(P)}$, где f_{w_i} — вероятность выбора w_i веса
$crossover\_point = random(0, len\,(p_1,\ p_2, \ldots,\ p_n))$
$new\_population := new\_population + w_i$
// Мутация:
for $i := 1$ **to** $new\_population$ **do**
    Замена случайного веса w_i в диапазоне $weight\_limit$
    $population := population + new\_population$
Возврат на оценку приспособленности

Для оценки эффективности предложенных алгоритмов проведен замер времени нахождения оптимальных весов для наборов модулей общего вида. Оптимальные веса для используемых в моделировании наборов модулей представлены в табл. 2.

Таблица 2. Веса, найденные алгоритмом 1 и 2

Набор модулей	Оптимальные веса	
	Метод Монте-Карло	Генетический алгоритм
$\{23, 29, 31\}$	$\{4, 17, 1\}$	$\{4, 17, 1\}$
$\{23, 29, 31, 37\}$	$\{7, 40, 11, 26\}$	$\{8, 3, 40, 37\}$
$\{23, 29, 31, 37, 41\}$	$\{7, 20, 27, 46, 48\}$	$\{14, 49, 22, 38, 10\}$
$\{23, 29, 31, 37, 41, 43\}$	$\{2, 25, 10, 11, 30, 38\}$	$\{1, 6, 27, 33, 27, 22\}$
$\{23, 29, 31, 37, 41, 43, 47\}$	$\{13, 18, 40, 26, 33, 6, 10\}$	$\{18, 6, 1, 12, 6, 19, 11\}$
$\{23, 29, 31, 37, 41, 43, 47, 53\}$	$\{15, 10, 36, 38, 23, 2, 43, 28\}$	$\{18, 29, 3, 48, 23, 18, 41, 11\}$

В табл. 3 представлено время работы алгоритмов.

Для наглядности на рис. 3 отображены результаты времени выполнения алгоритмов из табл. 3.

Таблица 3. Время нахождения оптимальных весов, с

Набор модулей	Метод Монте-Карло	Генетический алгоритм
$\{23, 29, 31\}$	0.014	0.006
$\{23, 29, 31, 37\}$	2	12
$\{23, 29, 31, 37, 41\}$	47	24
$\{23, 29, 31, 37, 41, 43\}$	125	31
$\{23, 29, 31, 37, 41, 43, 47\}$	189	42
$\{23, 29, 31, 37, 41, 43, 47, 53\}$	278	48

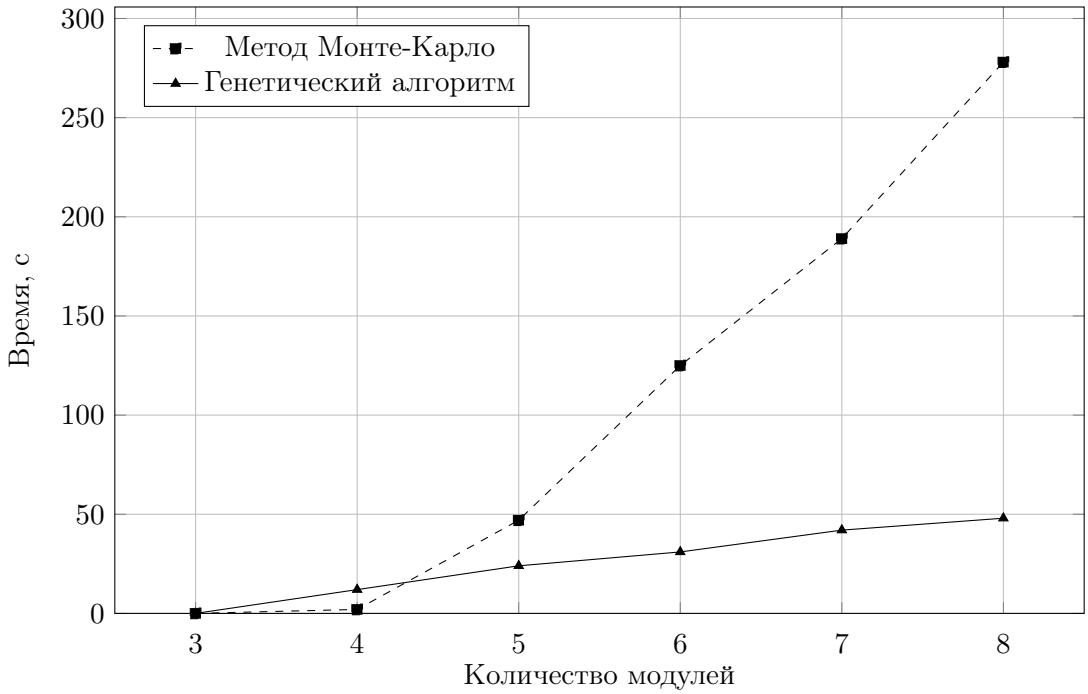

Рис. 3. График зависимости времени выполнения от количества модулей для методов подбора оптимальных весов

С увеличением размера набора модулей наблюдается значительное увеличение времени выполнения для обоих методов. Начиная с набора $\{23, 29, 31, 37\}$ и далее, генетический алгоритм демонстрирует значительно более высокую производительность по сравнению с методом Монте-Карло. Например, для набора из восьми модулей $\{23, 29, 31, 37, 41, 43, 47, 53\}$ генетический алгоритм работает примерно в 5.8 раз быстрее.

В среднем, генетический алгоритм быстрее метода Монте-Карло на 38%. Данные результаты связаны с тем, что метод Монте-Карло основан на случайной выборке и статистическом усреднении, он требует большого числа итераций для достижения точности, особенно в задачах высокой размерности [16]. Генетический алгоритм не просто случайно блуждает по пространству решений, а направленно улучшает популяцию решений через селекцию, скрещивание и мутацию. Для задач с большим числом модулей целесообразнее использовать генетический алгоритм, который показывает лучшее время выполнения, несмотря на усложнение задачи.

Для доказательства эффективности работы функции с оптимальными весами как позиционной характеристики, проведем следующее моделирования. Сравним время вычисления дружественных функций с функцией Пирло [20]. Функция Пирло имеет следующий вид

$$Pi\left(X\right) = \left| \sum_{i=1}^{n} k_i^{**} \cdot x_i \right|_{P_n}, \tag{9}$$

где $k_i^{**} = \left\lfloor \frac{\left|P_i^{-1}\right|_{p_i} P_i}{p_n} \right\rfloor$.

Рассмотрим пример вычисления функции Пирло.

Пример 5. Для числа $X = (2, 2, 3)$, представленное в СОК с модулями $\{3, 5, 7\}$ найдем позиционную характиристику с использование функции Пирло. Для начала вычислим необходимые константы:

$$\begin{aligned} Pi\left(P\right) &= P_n = P_3 = 15, \\ k_1^{**} &= \left\lfloor \frac{\left|P_1^{-1}\right|_{p_1} P_1}{p_3} \right\rfloor = \left\lfloor \frac{2 \cdot 35}{7} \right\rfloor = 10, \\ k_2^{**} &= \left\lfloor \frac{\left|P_2^{-1}\right|_{p_2} P_2}{p_3} \right\rfloor = \left\lfloor \frac{1 \cdot 21}{7} \right\rfloor = 3, \\ k_3^{**} &= \left\lfloor \frac{\left|P_3^{-1}\right|_{p_3} P_3}{p_3} \right\rfloor = \left\lfloor \frac{1 \cdot 15}{7} \right\rfloor = 2. \end{aligned}$$

Найдем значение функции Пирло

$$Pi\left(X\right) = \left|2 \cdot 10 + \ 2 \cdot 3 + 3 \cdot 2\right|_{15} = 2.$$

Отметим, что функция Пирло является частным случаем функции ядра Акушского с весами $\{0, 0, \ldots, w_{n-1} = 0, w_n = 1\}$.

Были использованы оптимальные веса, найденные генетическим методом из табл. 2. Для чистоты эксперимента, замер времени проводился только для нахождения $C\left(X\right)$ и $Pi\left(X\right)$. Остальные данные, необходимые для расчетов вычислены заранее. Позиционные характеристики вычислялись для максимального числа из диапазона $[0, P)$. Результаты замеров времени показаны в табл. 4.

Таблица 4. Время нахождения позиционных характеристик, мс

Набор модулей	Функция Пирло	Функция ядра
$\{23, 29, 31\}$	23	19
$\{23, 29, 31, 37\}$	25	22
$\{23, 29, 31, 37, 41\}$	26	23
$\{23, 29, 31, 37, 41, 43\}$	27	25
$\{23, 29, 31, 37, 41, 43, 47\}$	36	31
$\{23, 29, 31, 37, 41, 43, 47, 53\}$	39	35

В среднем, нахождение функции ядра Акушского с использованием оптимальных весов быстрее расчета функции Пирло на 14%.

Заключение

В работе рассмотрена функция ядра Акушского как универсальный инструмент для определения позиционной характеристики числа в системе остаточных классов.

С целью поиска оптимальных весов функции ядра Акушского в статье предложено использование метода Монте-Карло и генетического алгоритма. Экспериментальные результаты показали, что генетический алгоритм обеспечивает более стабильные результаты, при увеличении количества модулей, в то время как метод Монте-Карло быстрее на малых размерностях, но менее эффективен при усложнении задачи. В среднем генетический алгоритм быстрее метода Монте-Карло на 38%, что делает его предпочтительным для выбора.

Дополнительно был проведен эксперимент, в котором сравнивалось вычисление позиционной характеристики числа в СОК на основе функции ядра с оптимальными весами и функцией Пирло. Эксперимент показал, что вычисление функции ядра с оптимальными весами в среднем на 14% быстрее чем вычислении функции Пирло. Дальнейшие исследования будут направлены на использование функции ядра Акушского в криптографических алгоритмах.

Исследование выполнено за счет гранта Российского научного фонда № 24-21-00149, https://rscf.ru/project/24-21-00149/.

Литература

1. Schoinianakis D. Residue arithmetic systems in cryptography: a survey on modern security applications // Journal of Cryptographic Engineering. 2020. Vol. 10, no. 3. P. 249–267. DOI: 10.1007/s13389-020-00231-w.

2. Cardarilli G.C., Nunzio L.D., Fazzolari R. An RNS-based initial absolute position estimator for Electrical Encoders // IEEE Access. 2023. Vol. 11. P. 98586–98595. DOI: 10.1109/access.2023.3312619.

3. Mohan P.V.A. Residue number systems: algorithms and architectures. Springer Science & Business Media, 2002.

4. Omondi A.R., Premkumar A.B. Residue number systems: theory and implementation. World Scientific, 2007. Vol. 2.

5. Bajard J.C., Eynard J. RNS Approach in Lattice-Based Cryptography // Embedded Systems Design with Special Arithmetic and Number Systems. 2017. P. 345–368. DOI: 10.1007/978-3-319-49742-6_13.

6. Valueva M. V., Nagornov N.N., Lyakhov P.A., *et al.* Application of the residue number system to reduce hardware costs of the convolutional neural network implementation // Mathematics and Computers in Simulation. 2020. Vol. 177. P. 232–243. DOI: 10.1016/j.matcom.2020.04.031.

7. Roohi A., Taheri M.R., Angizi S., *et al.* Rnsim: Efficient deep neural network accelerator using residue number systems // 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE, 2021. P. 1–9. DOI: 10.1109/iccad51958.2021.9643531.

8. Lutsenko V., Babenko M., Deryabin M. Construction of Akushsky Core Functions Without Critical Cores // Mathematics. 2024. Vol. 12, no. 21. P. 3399. DOI: 10.3390/math12213399.

9. Lutsenko V.V., Babenko M.G., Khamidov M.M. High speed method of conversion numbers from residue number system to positional notation // Proceedings of the Institute for System Programming of the RAS. 2024. Vol. 36, no. 4. P. 117–132. DOI: 10.15514/ispras-2024-36(4)-9.

10. Shiriaev E., Kucherov N., Babenko M., *et al.* Fast operation of determining the sign of a number in RNS using the Akushsky core function // Computation. 2023. Vol. 11, no. 7. P. 124. DOI: 10.3390/computation11070124.

11. Lutsenko V., Geryugova A., Babenko M., *et al.* High-Speed Parity Number Detection Algorithm in RNS Based on Akushsky Core Function // International Conference on Communication and Computational Technologies / eds. by S. Kumar, S. Hiranwal, R. Garg, S.D. Purohit. Cham: Springer, Singapore, 2024. P. 491–504. DOI: 10.1007/978-981-97-7423-4_38.

12. Луценко В.В., Бабенко М.Г., Черных А.Н., Лапина М.А. Оптимизация алгоритма деления чисел в системе остаточных классов на основе функции ядра Акушского. Труды ИСП РАН. 2023. Т. 35, № 5. С. 157–168. DOI: 10.15514/ispras-2022-35(5)-11.

13. Wang Y. Residue-to-binary converters based on new Chinese remainder theorems // IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. 2000. Vol. 47, no. 3. P. 197–205. DOI: 10.1109/82.826745.

14. Акушский И.Я., Бурцев В.М., Пак И.Т. О новой позиционной характеристике непозиционного кода и ее приложении // Теория кодирования и оптимизация сложных систем. Алма-Ата, Наука, КазССР. 1977. С. 8–16.

15. Metropolis N., Ulam S. The Monte Carlo method // Journal of the American Statistical Association. 1949. Vol. 44, no. 247. P. 335–341.

16. Kroese D.P., Brereton T., Taimre T., *et al.* Why the Monte Carlo method is so important today // Wiley Interdisciplinary Reviews: Computational Statistics. 2014. Vol. 6, no. 6. P. 386–392. DOI: 10.1002/wics.1314.

17. Shiriaev E., Kucherov N., M Babenko M., *et al.* Algorithm for determining the optimal weights for the Akushsky core function with an approximate rank // Applied Sciences. 2023. Vol. 13, no. 18. P. 10495. DOI: 10.3390/app131810495.

18. Курейчик В.М. Генетические алгоритмы // Известия Южного федерального университета. Технические науки. 1998. Т. 8, № 2. С. 4–7.

19. Mirjalili S. Evolutionary algorithms and neural networks // Studies in Computational Intelligence. 2019. Vol. 780, no. 1. P. 43–53. DOI: 10.1007/978-3-319-93025-1.

20. Pirlo G., Impedovo D. A new class of monotone functions of the residue number system // International Journal of Mathematical Models and Methods in Applied Sciences. 2013. Vol. 7, no. 9. P. 802–809.

Луценко Владислав Вячеславович, инженер-исследователь, департамент науки, Северо-Кавказский федеральный университет (Ставрополь, Российская Федерация)

Горлачев Дмитрий Евгеньевич, студент, кафедра вычислительной математики и кибернетики, Северо-Кавказский федеральный университет (Ставрополь, Российская Федерация)

Мирный Никита Михайлович, студент, кафедра вычислительной математики и кибернетики, Северо-Кавказский федеральный университет (Ставрополь, Российская Федерация)

Бабенко Михаил Григорьевич, д.ф.-м.н., заведующий кафедрой вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова, Северо-Кавказский федеральный университет (Ставрополь, Российская Федерация)

DOI: 10.14529/cmse250202

SEARCHING OF OPTIMAL WEIGHTS FOR THE AKUSHSKY CORE FUNCTION

© 2025 V.V. Lutsenko, D.E. Gorlacev, N.M. Mirny, M.G. Babenko

North-Caucasus Federal University

(Pushkin st. 1, Stavropol, 355017 Russia)

E-mail: officialvladlutsenko@gmail.com, dimagorlacev12@gmail.com, mirnyjn@gmail.com, mgbabenko@ncfu.ru

Received: 07.03.2025

Modern computational tasks involving the processing of large numbers require not only high accuracy but also significant speed. In this context, the use of the residue number system offers an approach to parallel big data processing used in cryptography, signal processing and artificial neural networks. Despite the advantages of the residue number system, its diffusion has been slow due to the computational complexity of the so-called non-modular operations of the residual class system. One of the universal tools for realising non-modular operations is the Akushsky core function. This paper studies the Akushsky core function as a tool for determining the positional characteristic of a number in the residue number system. The application of Monte Carlo method and genetic algorithm is proposed to find the optimal weights of the core function. Experimental results demonstrate that the genetic algorithm provides more stable results when the number of moduli increases, while the Monte Carlo method is effective on small dimensions. The genetic algorithm is on average 38% faster than the Monte Carlo method, making it the preferred choice. Additionally, the computation time of the core function with optimal weights and the Pirlo function were compared. The results showed that the core function with optimal weights is on average 14% faster than the Pirlo function.

Keywords: residue number system, high-performance computing, Akushsky core function, Monte Carlo method, genetic algorithm.

FOR CITATION

Lutsenko V.V., Gorlacev D.E., Mirny N.M., Babenko M.G. Searching of Optimal Weights for the Akushsky Core Function. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2025. Vol. 14, no. 2. P. 26–41. (in Russian) DOI: 10.14529/cmse250202.

References

1. Schoinianakis D. Residue arithmetic systems in cryptography: a survey on modern security applications. Journal of Cryptographic Engineering. 2020. Vol. 10, no. 3. P. 249–267.

DOI: 10.1007/s13389-020-00231-w.

2. Cardarilli G.C., Nunzio L.D., Fazzolari R. An RNS-based initial absolute position estimator for Electrical Encoders. IEEE Access. 2023. Vol. 11. P. 98586–98595. DOI: 10.1109/access.2023.3312619.
3. Mohan P.V.A. Residue number systems: algorithms and architectures. Springer Science & Business Media, 2002.
4. Omondi A.R., Premkumar A.B. Residue number systems: theory and implementation. World Scientific, 2007. Vol. 2.
5. Bajard J.C., Eynard J. RNS Approach in Lattice-Based Cryptography. Embedded Systems Design with Special Arithmetic and Number Systems. 2017. P. 345–368. DOI: 10.1007/978-3-319-49742-6_13.
6. Valueva M.V., Nagornov N.N., Lyakhov P.A., *et al.* Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation. 2020. Vol. 177. P. 232–243. DOI: 10.1016/j.matcom.2020.04.031.
7. Roohi A., Taheri M.R., Angizi S., *et al.* Rnsim: Efficient deep neural network accelerator using residue number systems. 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE, 2021. P. 1–9. DOI: 10.1109/iccad51958.2021.9643531.
8. Lutsenko V., Babenko M., Deryabin M. Construction of Akushsky Core Functions Without Critical Cores. Mathematics. 2024. Vol. 12, no. 21. P. 3399. DOI: 10.3390/math12213399.
9. Lutsenko V.V., Babenko M.G., Khamidov M.M. High speed method of conversion numbers from residue number system to positional notation. Proceedings of the Institute for System Programming of the RAS. 2024. Vol. 36, no. 4. P. 117–132. DOI: 10.15514/ispras-2024-36(4)-9.
10. Shiriaev E., Kucherov N., Babenko M., *et al.* Fast operation of determining the sign of a number in RNS using the Akushsky core function. Computation. 2023. Vol. 11, no. 7. P. 124. DOI: 10.3390/computation11070124.
11. Lutsenko V., Geryugova A., Babenko M., *et al.* High-Speed Parity Number Detection Algorithm in RNS Based on Akushsky Core Function. International Conference on Communication and Computational Technologies / eds. by S. Kumar, S. Hiranwal, R. Garg, S.D. Purohit. Cham: Springer, Singapore, 2024. P. 491–504. DOI: 10.1007/978-981-97-7423-4_38.
12. Lutsenko V.V., Babenko M.G., Chernykh A.N., Lapina M.A. Optimization of a number division algorithm in the residue number system based on the Akushsky core function. Proceedings of ISP RAS. 2023. Vol. 35, no. 5. P. 157–168. (in Russian) DOI: 10.15514/ispras-2022-35(5)-11.
13. Wang Y. Residue-to-binary converters based on new Chinese remainder theorems. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. 2000. Vol. 47, no. 3. P. 197–205. DOI: 10.1109/82.826745.
14. Akushskiy I.Ya., Burtsev V.M., Pak I.T. About new positional characteristic of non-positional code and its application. Theory of Coding and Optimisation of Complex Systems. Alma-Ata, Nauka, KazSSR. 1977. P. 8–16. (in Russian)
15. Metropolis N., Ulam S. The Monte Carlo method. Journal of the American Statistical Association. 1949. Vol. 44. no. 247. P. 335–341.

16. Kroese D.P., Brereton T., Taimre T., *et al.* Why the Monte Carlo method is so important today. Wiley Interdisciplinary Reviews: Computational Statistics. 2014. Vol. 6, no. 6. P. 386–392. DOI: 10.1002/wics.1314.

17. Shiriaev E., Kucherov N., M Babenko M., *et al.* Algorithm for determining the optimal weights for the Akushsky core function with an approximate rank. Applied Sciences. 2023. Vol. 13, no. 18. P. 10495. DOI: 10.3390/app131810495.

18. Kureichik V.M. Genetic algorithms. Izvestiya Yuzhny Federal University. Technical Sciences. 1998. Vol. 8, no. 2. P. 4–7. (in Russian)

19. Mirjalili S. Evolutionary algorithms and neural networks. Studies in Computational Intelligence. 2019. Vol. 780, no. 1. P. 43–53. DOI: 10.1007/978-3-319-93025-1.

20. Pirlo G., Impedovo D. A new class of monotone functions of the residue number system. International Journal of Mathematical Models and Methods in Applied Sciences. 2013. Vol. 7, no. 9. P. 802–809.